



工业相机 SDK

示例程序和插件使用说明

在法律允许的最大范围内，本文档是“按照现状”提供，可能存在瑕疵或错误。本公司不对本文档提供任何形式的明示或默示保证，包括但不限于适销性、质量满意度、适合特定目的、不侵犯第三方权利等保证；亦不对使用或是分发本文档导致的任何特殊、附带、偶然或间接的损害进行赔偿，包括但不限于商业利润损失、系统故障、数据或文档丢失产生的损失。

目 录

Demo 更新记录.....	1
第 1 章 Demo 介绍.....	2
1.1 Demo 说明.....	2
1.2 获取途径.....	2
1.3 运行环境.....	3
第 2 章 基础 Demo 介绍.....	4
2.1 使用步骤.....	5
2.1.1 BasicDemo.....	5
2.1.2 ReconnectDemo.....	7
2.1.3 SetIODemo	8
2.1.4 ForcelpDemo.....	9
2.1.5 MultipleDemo	10
2.1.6 BasicDemoByGenTL	12
2.1.7 ReconstrcutImageDemo.....	13
2.1.8 ParameterGUIDemo.....	14
2.2 开发步骤.....	16
2.2.1 BCB	16
2.2.2 C#.....	17
2.2.3 VB	19
2.2.4 VC6.0	21
2.2.5 VS.....	23
2.2.6 XE5.....	25
第 3 章 第三方 Demo 介绍.....	27
3.1 Halcon 示例程序.....	27
3.1.2 使用步骤.....	27
3.1.3 开发步骤.....	29
3.2 Labview 示例程序	36
3.2.1 Labview VI	36
3.2.2 使用步骤.....	38

3.2.3 开发步骤.....	41
第 4 章 插件介绍.....	43
4.1 DirectShow	43
4.1.1 环境配置.....	43
4.1.2 使用说明.....	44
4.1.3 示例程序.....	47
4.2 Halcon.....	47
4.2.2 环境配置.....	47
4.2.3 相机配置.....	48
4.3 Sherlock.....	50
4.3.1 环境配置.....	50
4.3.2 相机配置.....	52
4.3.3 其他功能.....	54

Demo 更新记录

版本号	更新记录
4.0.0	<ul style="list-style-type: none">● 提供 BCB、C#、VB、VC6.0、VS 和 XE5 六种编译环境 Demo。● 提供 Halcon 和 Labview 两种第三方软件的 Demo。● 提供 DirectShow、Halcon 和 Sherlock 三个插件。

第1章 Demo 介绍

1.1 Demo 说明

工业相机 SDK 提供多种编译环境 Demo，方便用户进行二次开发，分为基础 Demo 和第三方 Demo 两种。除此之外，还提供多种插件，用于连接工业相机。

- 基础 Demo 涉及的平台为 BCB、C#、VB、VC6.0、VS、XE5，具体请见第 2 章 基础 Demo 介绍章节。
- 第三方 Demo 涉及的平台为 Halcon、Labview，具体请见第 3 章 第三方 Demo 介绍章节。
- 插件涉及 Halcon、DirectShow、Sherlock，具体请见第 4 章 插件介绍章节。

1.2 获取途径

工业相机 SDK 二次开发相关资料已集成在客户端中，安装后可在安装路径下找到。

操作步骤

1. 选中桌面上的客户端快捷方式。
2. 右键单击选择“打开文件所在的位置”，进入程序安装路径。
3. 从“Applications”文件夹回退到客户端安装目录，进入“Development”即可。
4. 根据实际需求选择对应版本 SDK 文件夹并进入，该路径下各文件夹存放对应的 SDK 资料，具体介绍请见下表。

表1-1 二次开发文件夹介绍

文件夹名称	内容
Bin	示例程序的可执行程序
Documentations	二次开发说明文档
DotNet	.Net 库
Includes	头文件
Libraries	链接库
Samples	示例程序代码
ThirdPartyPlatformAdapter	第三方平台插件

1.3 运行环境

- 操作系统：Windows XP（32 位中、英文操作系统），Windows 7/10（32/64 位中、英文操作系统），Windows Server（32/64 位中、英文操作系统）
- .NET 运行环境：.NET3.5 及以上

第2章 基础 Demo 介绍

工业相机 SDK 提供多种开发语言以及多种功能的基础 Demo，方便用户进行二次开发。开发语言涉及 C++、C#、VB、VC 等，不同开发语言 Demo 的界面略有差异，但功能基本一致，具体请以实际 Demo 功能为准。

Demo 可分为 MFC Demo 和控制台 Demo，各 MFC Demo 主要功能、支持的编译环境请见下表。



说明

本文档仅对 MFC Demo 做简要介绍，控制台 Demo 操作步骤可参考 MFC Demo。

表2-1 Demo 介绍

Demo	编译环境	功能说明
BasicDemo	BCB、C#、VB、VC60、VS、XE5	包含 SDK 使用过程中常用的接口调用，涵盖大多数用户对 SDK 的使用方法示例需求 说明 初次使用时，推荐参考 BasicDemo。
ReconnectDemo	C#、VB、VC60、VS	主要说明 SDK 中相机断线重连操作
SetIODemo	C#、VB、VC60、VS	主要说明 SDK 中对相机 IO 输入输出的控制操作
ForcelpDemo	C#、VB、VC60、VS	主要说明 SDK 中手动设置网口相机 IP 地址操作
MultipleDemo	C#、VB、VC60、VS	主要说明如何同时对多台相机进行一系列基本操作
BasicDemoByGenTL	C#、VC60、VS	主要说明如何通过加载 cti 文件的方式枚举并控制相机
ReonstrcutImageDemo	VC60、VS	主要说明如何对采集到的图像进行图像重组
ParameterGUIDemo	VC60、VS	主要说明如何使用 SDK 中的属性配置可视化控件

2.1 使用步骤

2.1.1 BasicDemo

BasicDemo 为基本示例程序，包含 SDK 使用过程中常用的一些接口调用。初次使用工业相机 SDK 进行二次开发时，推荐首先参考 BasicDemo。

BasicDemo 界面如下图所示，不同编译环境下的 BasicDemo 界面略有差异，但功能基本一致，各区域功能介绍请见下表。

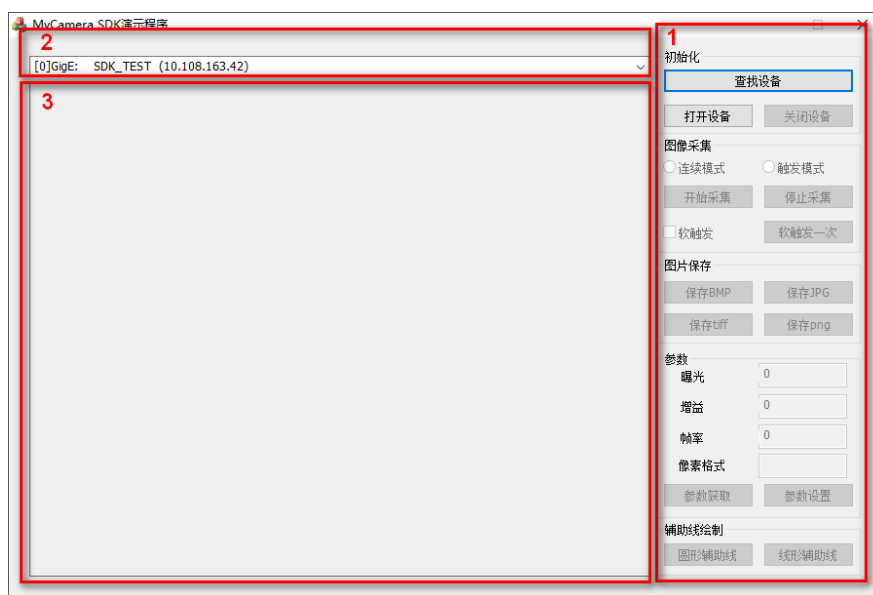


图2-2 BasicDemo

表2-2 BasicDemo 功能说明

编号	名称	功能说明
1	控制模块	可进行初始化、图像采集、图片保存、参数设置等操作。
2	下拉设备列表	下拉可显示设备进行连接。
3	图像显示区域	可显示已采集到的图像。

操作步骤

1. 点击“查找设备”，此时下拉设备列表处会出现当前在线的设备列表，选择其中一个设备即可。

设备命名方式有如下两种情况：

- 设备名称不为空时，下拉列表处显示设备类型+设备名称+IP 地址；

- 设备名称为空时，下拉列表处显示设备类型+设备型号+IP 地址。

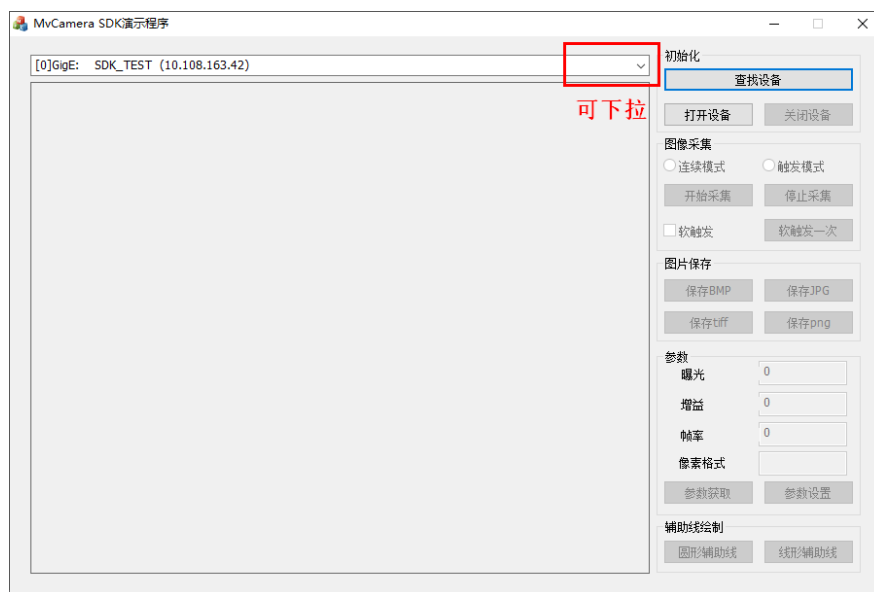


图2-3 查找设备

2. 点击“打开设备”，可打开当前选中的设备，打开设备后默认以连续方式取流。
3. 通过控制模块下的“图像采集”，可选择“连续模式”或“触发模式”，如下图所示。



图2-4 连续或触发模式

- 选择“触发模式”时，可以设置为软触发。点击“开始采集”后，可通过点击“软触发一次”完成触发一次功能。
- 选择“连续模式”时，点击“开始采集”进行图像采集，图像显示区域出现实时图像。

4. (可选) 若想保存当前图像, 点击“图片保存”下“保存 BMP”、“保存 JPG”、“保存 tiff”或者“保存 png”, 即可将对应类型的图片保存在当前 exe 目录下。
5. (可选) 通过控制模块下的“参数”, 可获取参数或设置参数。
 - 点击“参数获取”获得当前的曝光时间、增益和帧率。
 - 更改“曝光”、“增益”和“帧率”的数值, 点击“参数设置”即可生效。



图2-5 设置参数

6. (可选) 点击“圆形辅助线”或“线形辅助线”, 可在图像显示区域绘制多个圆形或线形。



Basic Demo 还可绘制多个矩形, 但界面上未显示“矩形辅助线”按钮, 可通过调用接口实现。

2.1.2 ReconnectDemo

ReconnectDemo 主要展示 SDK 中如何使用断线回调并重新连接相机。

ReconnectDemo 界面如下图所示, 不同编译环境下的 ReconnectDemo 界面略有差异, 但功能基本一致, 各区域功能介绍请见下表。

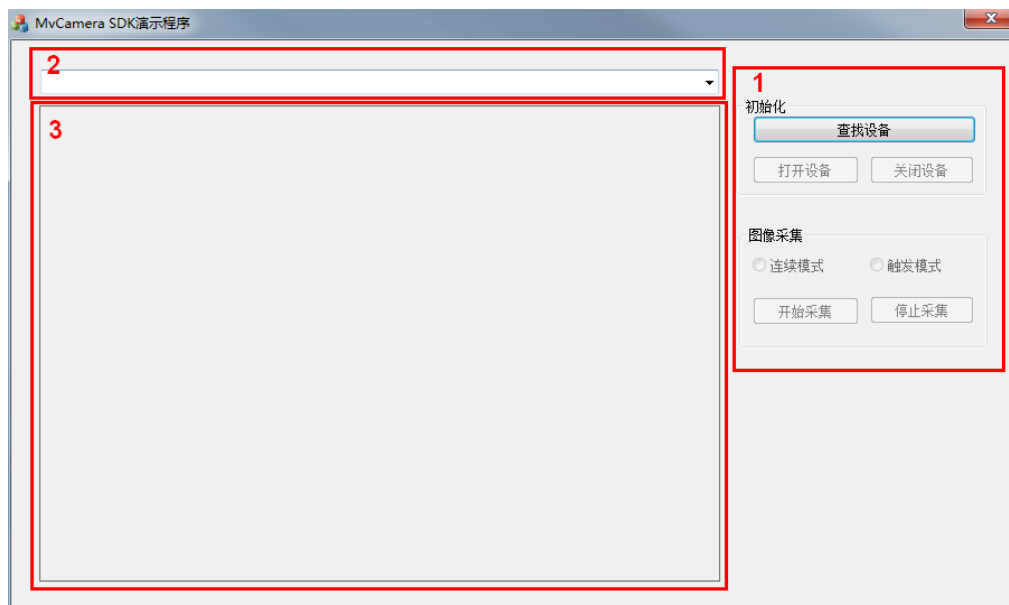


图2-6 ReconnectDemo

表2-3 ReconnectDemo 功能说明

编号	名称	功能说明
1	控制模块	可进行初始化、图像采集等操作。
2	下拉设备列表	下拉可显示设备进行连接。
3	图像显示区域	可显示已采集到的图像。

设备基本操作步骤与 BasicDemo 相似，具体请见 2.1.1 BasicDemo 章节。打开设备后，当设备异常断线，ReconnectDemo 进入异常回调程序，根据当前打开的设备信息不断的尝试连接，设备在线后将会被自动重新连接。

2.1.3 SetIODemo

SetIODemo 主要展示如何对相机 IO 进行输入输出控制。

SetIODemo 界面如下图所示，不同编译环境下的 SetIODemo 界面略有差异，但功能基本一致，各区域功能介绍请见下表。

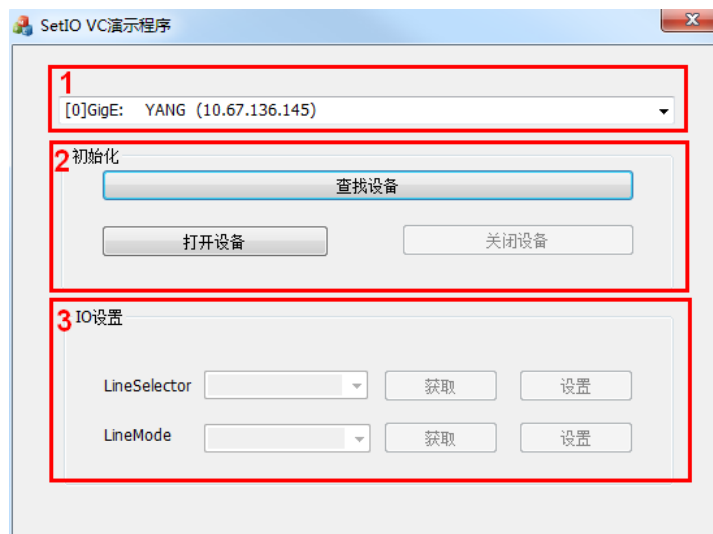


图2-7 SetIODemo

表2-4 SetIODemo 功能说明

编号	名称	功能说明
1	下拉设备列表	下拉可显示设备进行连接。
2	初始化模块	可进行查找设备、打开设备、关闭设备。
3	IO 设置模块	可对 IO 的输入输出进行简单设置。

操作步骤

1. 基本操作步骤与 BasicDemo 相似，具体请见 2.1.1 BasicDemo 章节。
2. 点击“LineSelector”下的“获取”，得到当前可选择的设备 IO 端口：Line0、Line1、Line2。
3. 选择需要设置的 IO 端口，点击“LineSelector”下的“设置”即可。
4. 点击“LineMode”下的“获取”，得到当前选择的 IO 端口的输入输出模式。
5. 选择需要设置的输入输出模式，点击“LineMode”下的“设置”即可。
 - 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。

2.1.4 ForceIpDemo

ForceIpDemo 主要展示如何对设备 IP 进行设置。

ForcelpDemo 界面如下图所示，不同编译环境下的 ForcelpDemo 界面略有差异，但功能基本一致，各区域功能介绍请见下表。



图2-8 ForcelpDemo

表2-5 ForcelpDemo 功能说明

编号	名称	功能说明
1	下拉设备列表	下拉可显示设备进行连接。
2	初始化模块	可进行查找设备、打开设备、关闭设备操作。
3	设置 IP 模块	可对设备 IP 进行设置。

操作步骤

1. 点击“查找设备”，对网段内的设备进行枚举，下拉选择需要配置 IP 的设备。
2. 根据“推荐 IP 范围”，在“IP 地址”中输入想要设置的 IP。
3. 点击“设置 IP”。

2.1.5 MultipleDemo

MultipleDemo 主要展示如何同时对多相机进行常用接口的调用。

MultipleDemo 界面如下图所示，不同编译环境下的 MultipleDemo 界面略有差异，但功能基本一致，各区域功能介绍请见下表。

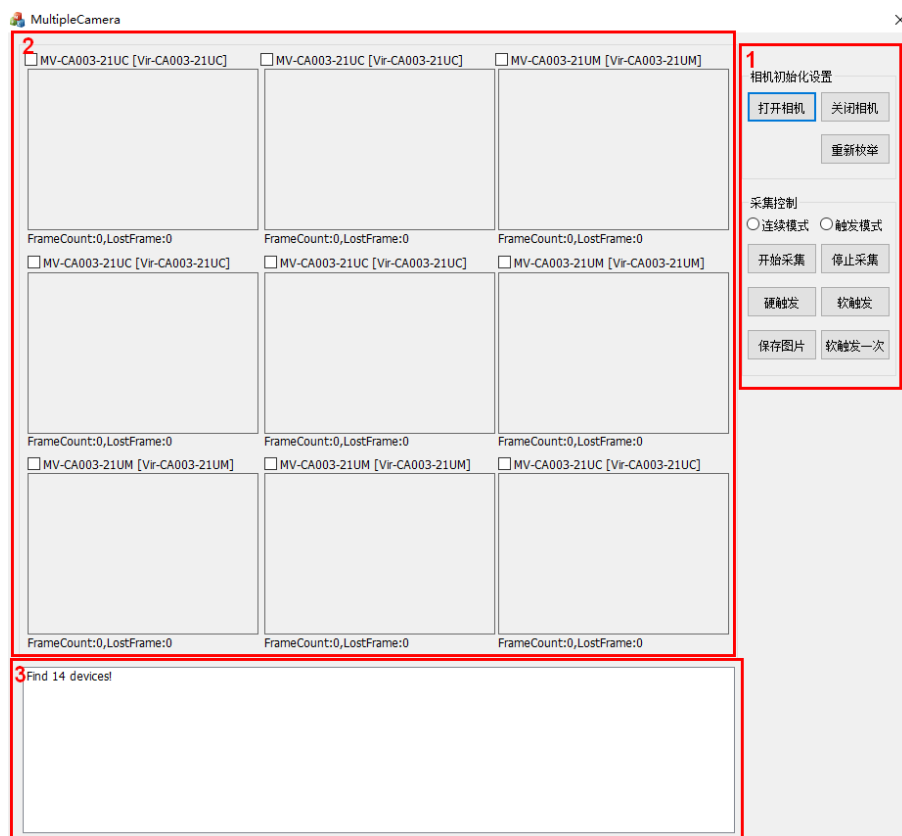


图2-9 MultipleDemo

表2-6 MultipleDemo 功能说明

编号	名称	功能说明
1	控制模块	可进行相机初始化、图像采集、图片保存等操作。
2	图像显示区域	可显示已采集到的图像，并在图像下方同步显示采集帧率和丢帧数。
3	操作信息输出框	可显示 Demo 运行过程的中的每步操作信息。

操作步骤

1. 打开软件后，自动枚举当前网段的相机，并将最前面的九个相机对应到相应窗口中。
2. 在图像显示区域勾选需要的相机，点击“打开相机”。
3. 通过控制模块下的“采集控制”，可选择“连续模式”或“触发模式”。
 - 选择“触发模式”时，可点击“硬触发”、“软触发”或“软触发一次”。
 - 选择“连续模式”时，点击“开始采集”进行图像采集，图像显示区域出现实时图像。同时采集帧数和丢帧数会即时更新数据（1 秒更新一次）。

4. （可选）若想保存当前图像，点击“保存图片”后，图像即可保存在当前 exe 目录下。
5. 点击“停止采集”或“关闭设备”即可停止采集图像。

2.1.6 BasicDemoByGenTL

BasicDemoByGenTL 主要展示如何基于 GenTL 标准加载不同的 CTI 文件，枚举到对应的设备，调用常用接口。

BasicDemoByGenTL 界面如下图所示，不同编译环境下的 BasicDemoByGenTL 界面略有差异，但功能基本一致，各区域功能介绍请见下表。

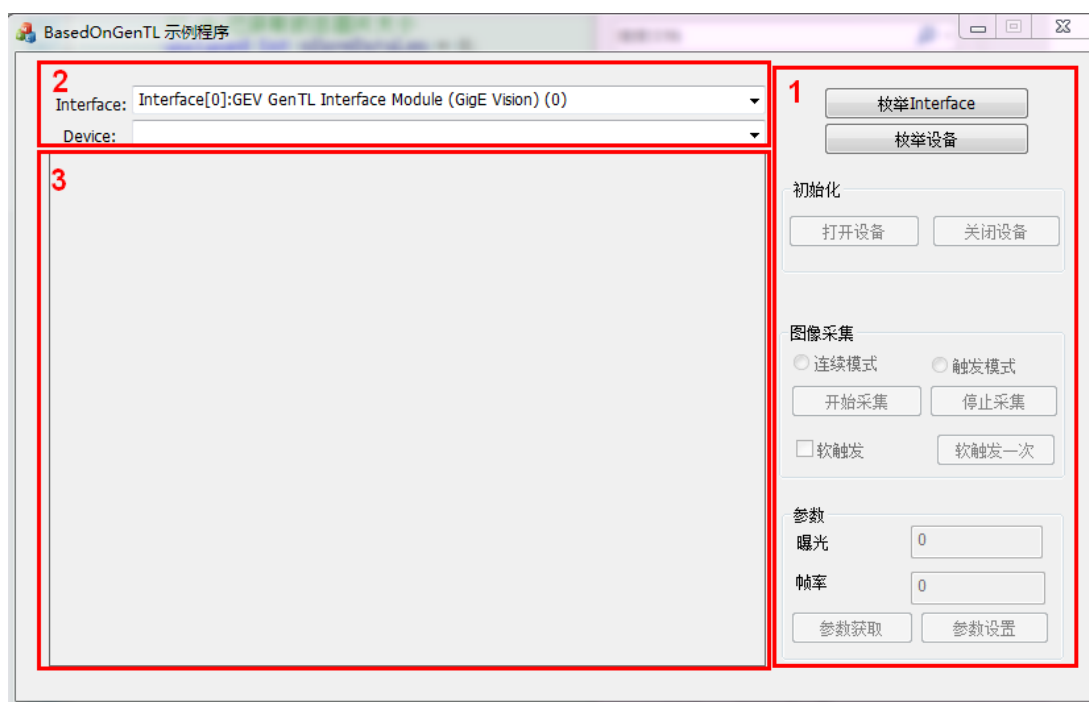


图2-10 BasicDemoByGenTL

表2-7 BasicDemoByGenTL 功能说明

编号	名称	功能说明
1	控制模块	可进行枚举 Interface、查找设备、初始化、图像采集、参数控制等操作。
2	下拉设备列表	下拉可显示设备进行连接。
3	图像显示区域	可显示已采集到的图像。

操作步骤

1. 点击“枚举 Interface”，选择需要的 CTI 文件。

2. 点击“枚举设备”进行查找设备，此时下拉设备列表处出现当前在线 GigEVision 设备列表，选择其中一个设备，如下图所示。



图2-11 枚举设备



说明

选择不同的 CTI 文件，会枚举出不同类型的设备。

3. 打开设备后，基本操作步骤与 BasicDemo 相似，具体请见 2.1.1 BasicDemo 章节。

2.1.7 ReconstrcutImageDemo

ReconstrcutImageDemo 主要展示如何进行相机图像重组操作，即按行拆分并拼接为多张图像，实际应用中可与线阵相机搭配，同时显示相机多个不同曝光值下的图像。

该示例程序仅演示功能，所以最多只能同时显示 2 张图像，实际应用中可支持最多同时显示 8 张图像，显示图像的个数由用户自行定义，如线阵相机分时曝光功能，可根据相机曝光个数节点确定同时显示的图像个数。

ReconstrcutImageDemo 界面如下图所示，不同编译环境下的 ReconstrcutImageDemo 界面略有差异，但功能基本一致，各区域功能介绍请见下表。

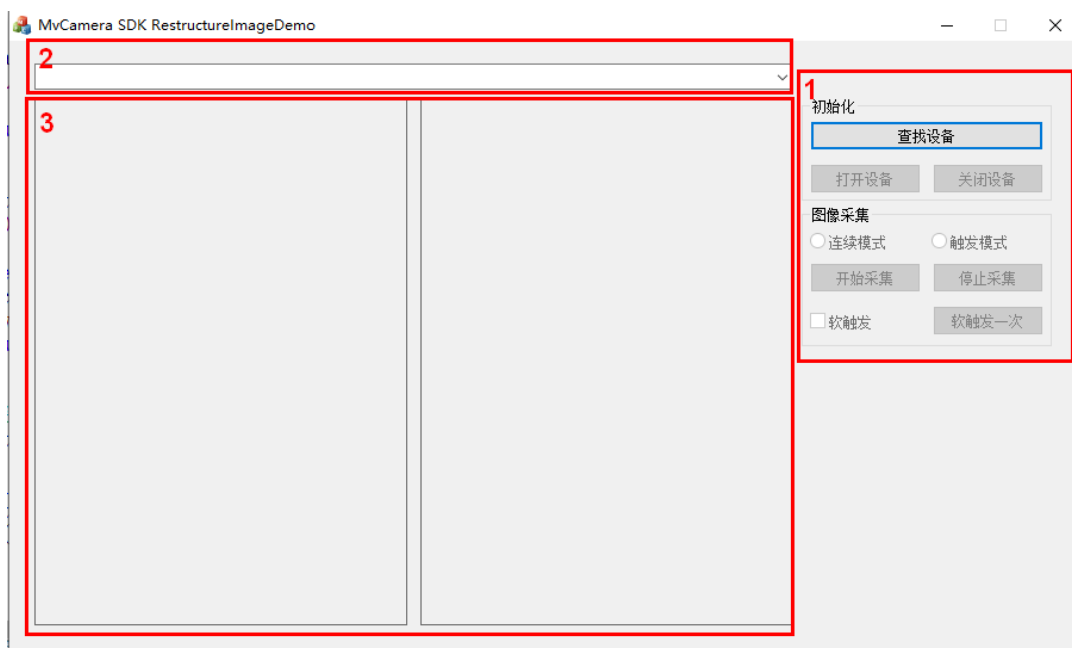


图2-12 ReconstructImageDemo

表2-8 ReconstructImageDemo 功能说明

编号	名称	功能说明
1	控制模块	可进行查找设备、打开设备、关闭设备、开始采集、停止采集、设置触发等操作。
2	下拉设备列表	下拉可显示设备进行连接。
3	图像显示区域	可显示已采集到的图像。

设备基本操作步骤与 BasicDemo 相似，具体请见 2.1.1 BasicDemo 章节。打开设备后，ReconstructImageDemo 将采集到的图像按行拆分成多张图像并同时显示在图像显示区域，可以清晰的对比每张图像。

2.1.8 ParameterGUIDemo

ParameterGUIDemo 主要展示如何使用属性配置控件，工业相机 SDK4.0 版本支持用户仅调用一个 SDK 接口就可以自动打开一个指定相机的属性配置可视化界面，该属性配置 GUI 界面可以修改相机的所有属性节点，可减少二次开发时间。

ParameterGUIDemo 界面如下图所示，不同编译环境下的 ParameterGUIDemo 界面略有差异，但功能基本一致，各区域功能介绍请见下表。

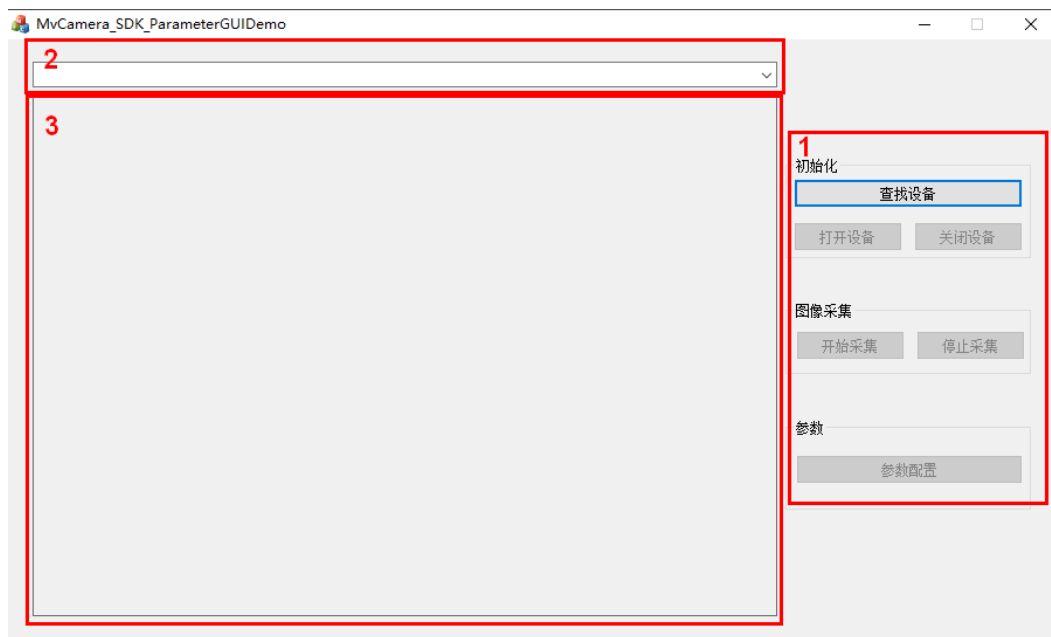


图2-13 ParameterGUIDemo

表2-9 ParameterGUIDemo 功能说明

编号	名称	功能说明
1	控制模块	可进行查找设备、打开设备、关闭设备、开始采集、停止采集、属性配置等操作。
2	下拉设备列表	下拉可显示设备进行连接。
3	图像显示区域	可显示已采集到的图像。

操作步骤

1. 设备基本操作步骤与 BasicDemo 相似，具体请见 2.1.1 BasicDemo 章节；
2. 点击“参数配置”，弹出已打开相机的属性配置界面，如下图所示。

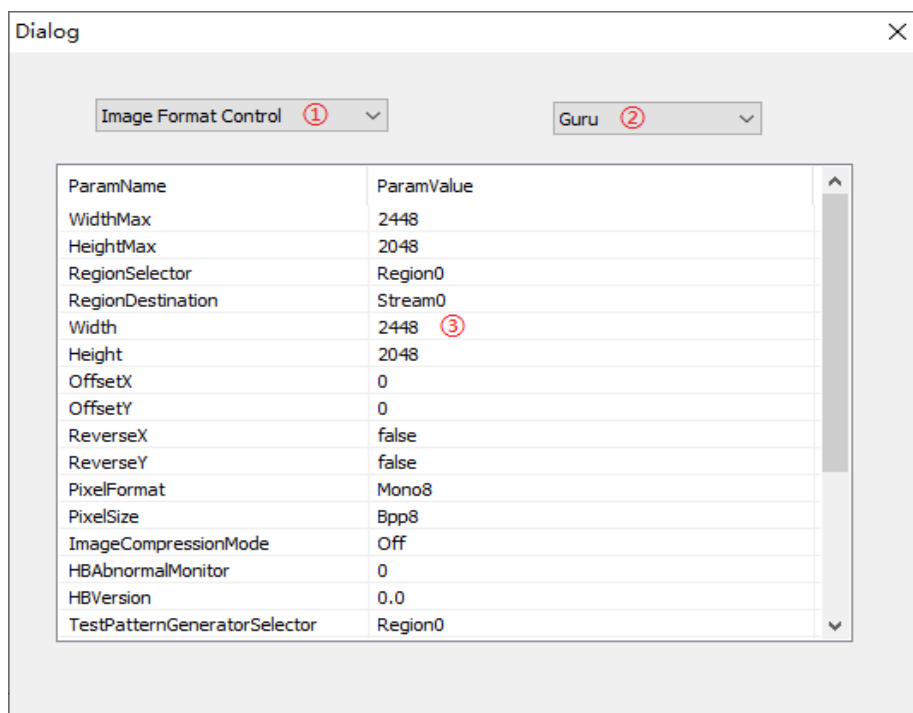


图2-14 属性配置界面

3. 通过界面右上角，下拉选择相机的节点显示权限，不同权限下显示的节点有所不同，用户可按需选择。

三种节点显示权限分别为：Beginner：初级；Expert：专家；Guru：大师。

4. 通过界面左上角，下拉选择相机主节点，相当于客户端中的属性目录。
5. 界面下方显示指定节点下的所有子节点，双击 ParamValue 列对需要修改的指定属性节点进行修改。

2.2 开发步骤

2.2.1 BCB

工业相机 SDK 可对接 BCB 软件，在 SDK 开发目录下，提供 1 个 MFC 示例程序。开发时需创建工程并配置头文件和 lib 目录，如图 2-15 和图 2-16 所示。

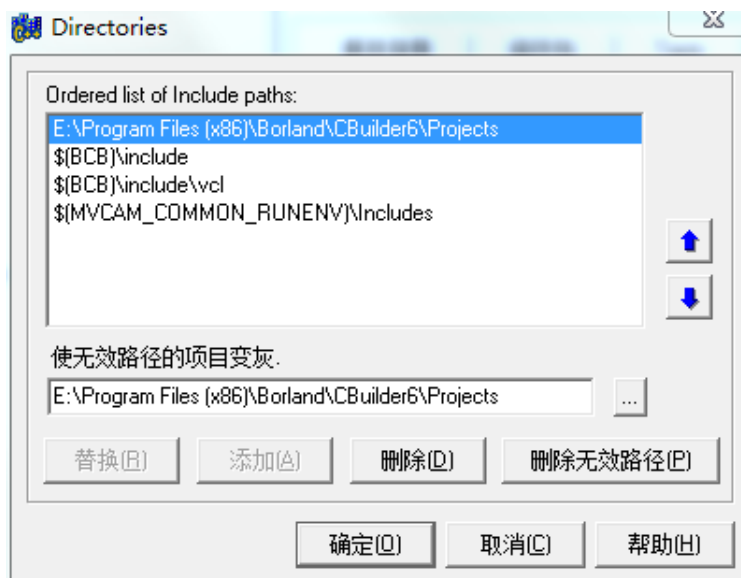


图2-15 配置头文件

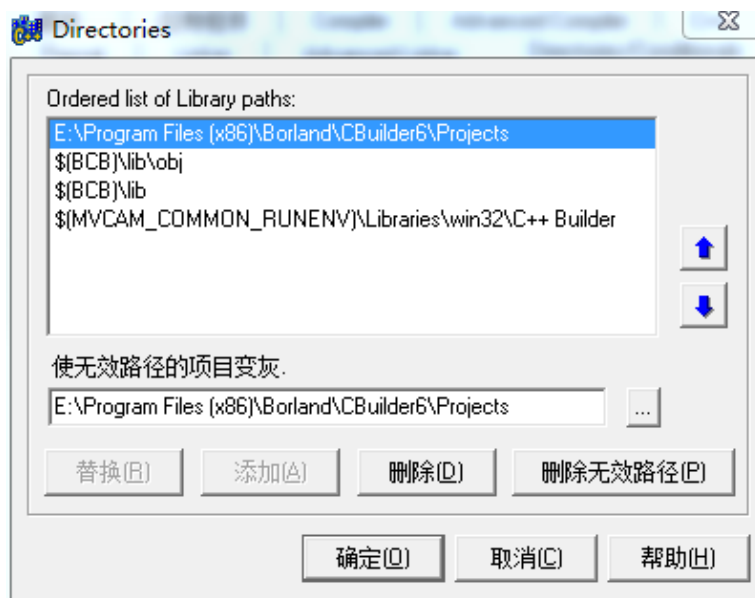


图2-16 配置 lib 目录

说明

- MVS 安装成功的同时，32 位和 64 位的 dll 文件自动被写入 PC 环境变量中。
- BCB Demo 与 C++程序链接使用的静态库不同，但函数接口一致，可调用相同头文件。

2.2.2 C#

使用工业相机 SDK 可开发 C#程序，在 SDK 开发目录下，提供 24 个 C#示例程序，其中 6 个为 MFC 程序，18 个为控制台程序。本文档仅对 MFC 程序的开发步骤做简要介绍。



说明

C#示例程序兼容中英文，对关键的程序均有中英文的注释，界面控件均有中英文区分，可通过 MFC 属性界面的 language 进行切换，如下图所示。

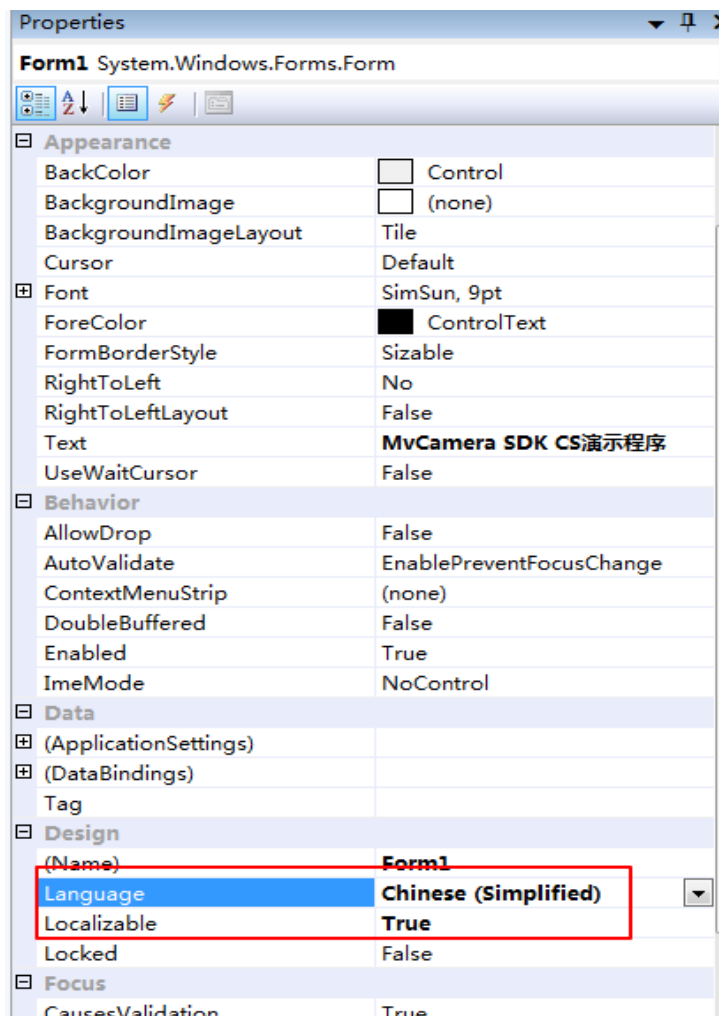


图2-17 语言切换

前提条件

客户端安装成功的同时，32 位、64 位和 AnyCpu 的 dll 文件自动被写入 PC 环境变量。

操作步骤

1. 创建 C#工程，在工程中将 MVCAM_COMMON_RUNENV\DotNet\AnyCpu 路径下的 MvCamCtrl.Net.dll 添加到工程引用中，如下图所示，MVCAM_COMMON_RUNENV 即为 PC 环境变量；

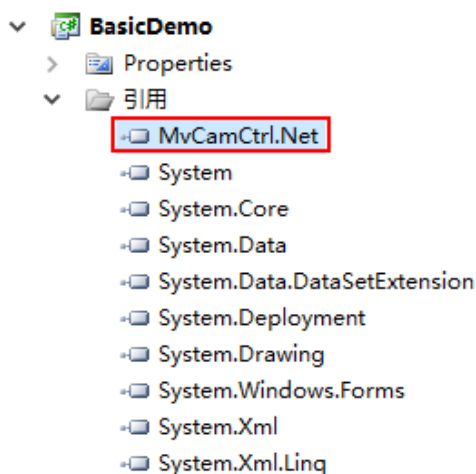


图2-18 工程配置

2. 引用命名空间 MvCamCtrl.NET，如下图所示，调用各个模块中相机操作函数。

```
...public class CSystem
{
    ...public const int MV_1394_DEVICE = 2;
    ...public const int MV_CAMERALINK_DEVICE = 8;
    ...public const int MV_GENTL_GIGE_DEVICE = 64;
    ...public const int MV_GIGE_DEVICE = 1;
    ...public const int MV_UNKNOWN_DEVICE = 0;
    ...public const int MV_USB_DEVICE = 4;
    ...public const int MV_VIR_GIGE_DEVICE = 16;
    ...public const int MV_VIR_USB_DEVICE = 32;

    public CSystem();

    ...public static int EnumCameraListByGenTL(ref CGenTLIFInfo pcIFInfo, ref List<CGenTLDevInfo> ltCameraList);
    ...public static int EnumDevices(uint nType, ref List<CCameraInfo> ltCameraList);
    ...public static int EnumDevicesEx(uint nType, ref List<CCameraInfo> ltCameraList, string strManufacturerName);
    ...public static int EnumDevicesEx2(uint nType, ref List<CCameraInfo> ltCameraList, string strManufacturerName, MV_SORT_METHOD enSortMethod);
    ...public static int EnumerateTIs();
    ...public static int EnumInterfaceByGenTL(ref List<CGenTLIFInfo> ltInterfaceList, string strGenTLPath);
    ...public static string GetSDKVersion();
    ...public static bool IsDeviceAccessible(ref CCameraInfo pcDevInfo, MV_ACCESS_MODE enAccessMode);
}
```

图2-19 引用命名空间

2.2.3 VB

使用工业相机 SDK 可开发 VB.NET 程序，在 SDK 开发目录下，提供 15 个 VB.NET 示例程序，其中 5 个为 MFC 程序，10 个为控制台程序。本文档仅对 MFC 程序的开发步骤做简要介绍。



说明

VB 示例程序兼容中英文，对关键的程序均有中英文的注释，界面控件均有中英文区分，可通过 MFC 属性界面的 language 进行切换，如下图所示。

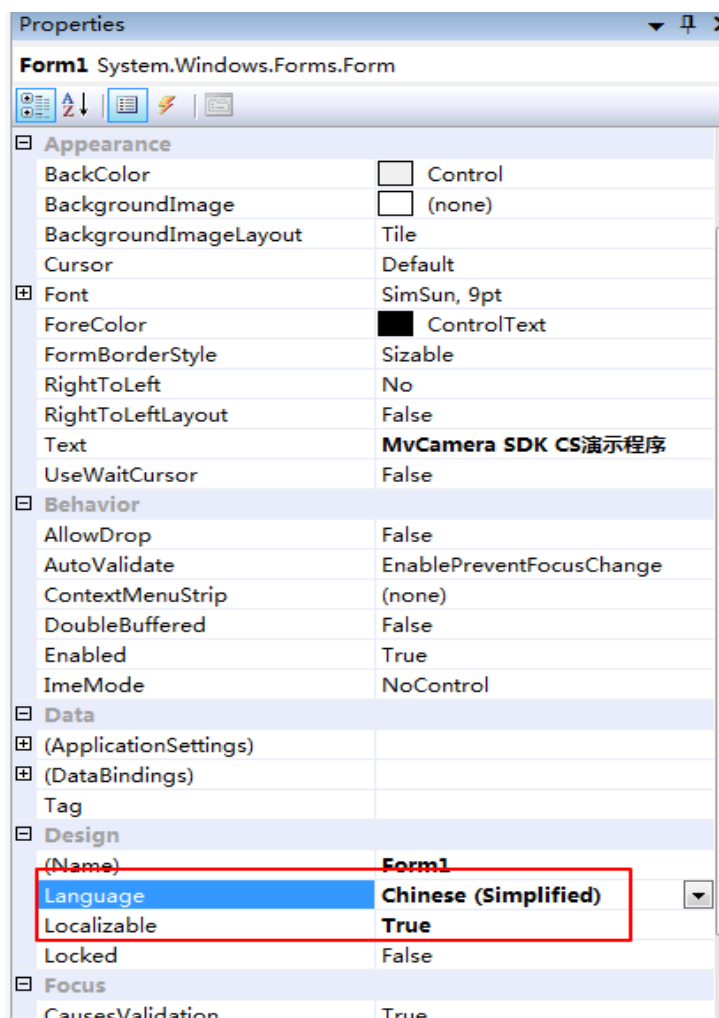


图2-20 语言切换

前提条件

客户端安装成功的同时，32 位、64 位和 AnyCpu 的 dll 文件自动被写入 PC 环境变量。

操作步骤

1. 由于 VBDemo 使用的是旧 C#库, SDK4.0 版本将旧 C#库放入 VB 示例程序同级目录下，导入同级目录下的 MvCameraControl.Net.dll 即可，如下图所示。

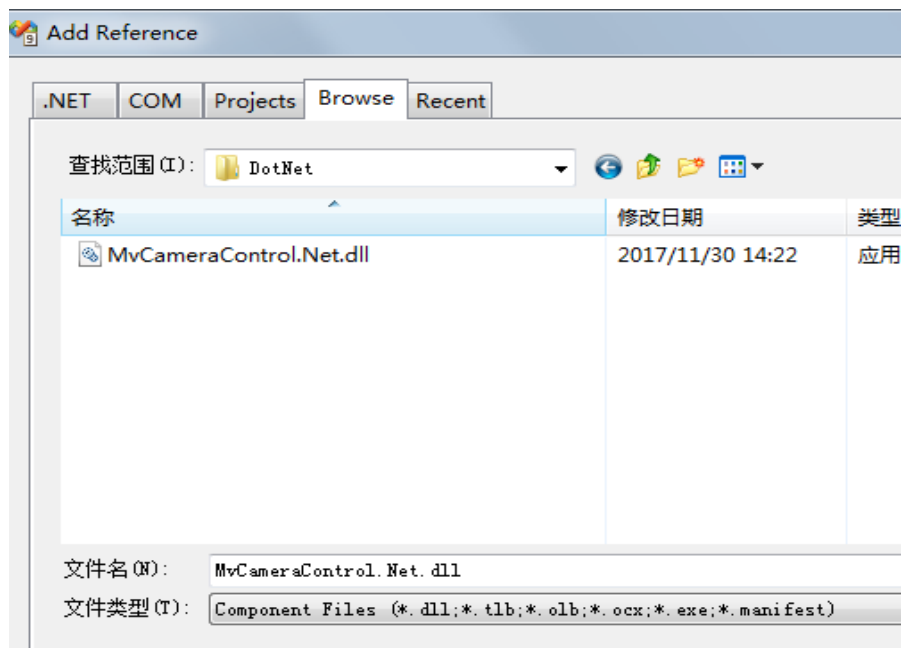


图2-21 导入.NET 库

2. 创建 MyCamera 类，如下图所示，该类包含 SDK 中关于相机的各种操作。

```
Imports System.Runtime.InteropServices
Imports System.Threading.Thread
Imports System.Net.IPAddress
Imports MvCamCtrl.NET

Public Class Thread
    ... Dim dev As MyCamera = New MyCamera
```

图2-22 创建 MyCamera 类

3. 可根据每个函数的介绍以及自己的需求进行二次开发。

2.2.4 VC6.0

使用工业相机 SDK 可开发 C++ 程序，在 SDK 开发目录下，提供 14 个 VC6.0 示例程序，其中 5 个为 MFC 程序，9 个为控制台程序。本文档仅对 MFC 程序的开发步骤做简要介绍。



说明

C++ 示例程序兼容中英文，对关键的程序均有中英文的注释，界面控件均有中英文区分，跟随系统语言自动切换。

前提条件

客户端安装成功的同时，32 位、64 位和 AnyCpu 的 dll 文件自动被写入 PC 环境变量。

操作步骤

1. 创建 VS 工程并添加引用，加入 MvSdkExport.lib 和 MvSdkExport.h 到工程中，如图 2-24 和图 2-23 所示；

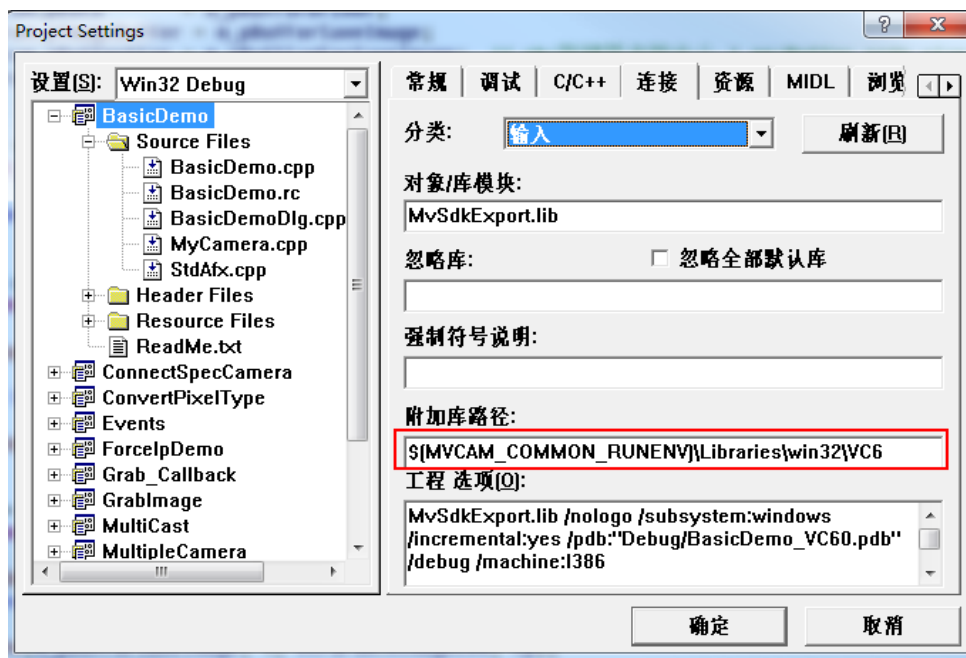


图2-23 加入 MvSdkExport.lib 文件

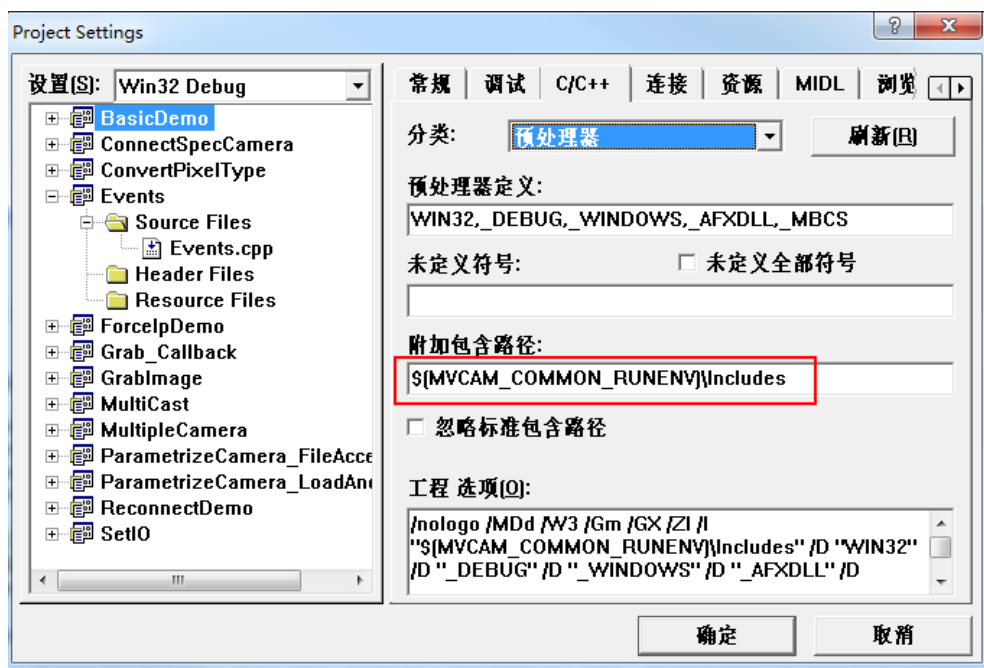


图2-24 加入 MvSdkExport.h 文件

2. 添加头文件和库文件引用，调用 MvSdkExport.h 中相机操作的函数，如下图所示。

```

/* ===== */
/* 相机的基本指令和操作 */
/* ===== */

HWSDKSPIR_API unsigned int _stdcall HW_CC_GetSDKVersion();

HWSDKSPIR_API int _stdcall HW_CC_EnumerateFis();

HWSDKSPIR_API int _stdcall HW_CC_EnumDevices(IN unsigned int nLayerType, IN OUT HW_CC_DEVICE_INFO_LIST *pstDevList);
HWSDKSPIR_API int _stdcall HW_CC_EnumDevices(IN unsigned int nLayerType, IN OUT HW_CC_DEVICE_INFO_LIST *pstDevList, IN const char * pManufacturerName);
HWSDKSPIR_API bool _stdcall HW_CC_IsDeviceAccessible(IN HW_CC_DEVICE_INFO *pstDeviceInfo, IN unsigned int nAccessMode);
HWSDKSPIR_API int _stdcall HW_CC_SetSDKLogPath(IN const char * pSDKLogPath);
HWSDKSPIR_API int _stdcall HW_CC_CreateHandle(IN void ** handle, IN const HW_CC_DEVICE_INFO * pstDeviceInfo);
HWSDKSPIR_API int _stdcall HW_CC_CreateHandleEx(IN void ** handle, IN const HW_CC_DEVICE_INFO * pstDeviceInfo);
HWSDKSPIR_API int _stdcall HW_CC_DestroyHandle(IN void * handle);
HWSDKSPIR_API int _stdcall HW_CC_OpenDeviceC(IN void * handle, IN unsigned int nAccessMode, IN unsigned short nSwitchOverKey);
HWSDKSPIR_API int _stdcall HW_CC_CloseDeviceC(IN void * handle);
HWSDKSPIR_API int _stdcall HW_CC_RegisterImageCallback(void *handle, void _stdcall cbOutput)(unsigned char * pData, HW_FRAME_OUT_INFO_EX pFrameInfo, void *pUser, void **pUser);
HWSDKSPIR_API int _stdcall HW_CC_RegisterImageBufferCB(void *handle, void _stdcall cbOutput)(unsigned char * pData, HW_FRAME_OUT_INFO_EX pFrameInfo, void **pUser, void **pUser);
HWSDKSPIR_API int _stdcall HW_CC_RegisterImageBlockCB(void *handle, void _stdcall cbOutput)(unsigned char * pData, HW_FRAME_OUT_INFO_EX pFrameInfo, void **pUser, void **pUser);
HWSDKSPIR_API int _stdcall HW_CC_StartGrabbing(IN void * handle);
HWSDKSPIR_API int _stdcall HW_CC_StopGrabbing(IN void * handle);
HWSDKSPIR_API int _stdcall HW_CC_SetParameter(IN void * handle, IN unsigned char * pData, IN unsigned int nDataSize, IN OUT HW_FRAME_OUT_INFO_EX pFrameInfo, int nEffect);
HWSDKSPIR_API int _stdcall HW_CC_GetParameter(RGB)(IN void * handle, IN unsigned char * pData, IN unsigned int nDataSize, IN OUT HW_FRAME_OUT_INFO_EX pFrameInfo, int nEffect);
HWSDKSPIR_API int _stdcall HW_CC_SetGammaCorrection(IN void * handle, IN unsigned char * pData, IN unsigned int nDataSize, IN OUT HW_FRAME_OUT_INFO_EX pFrameInfo, unsigned int nEffect);
HWSDKSPIR_API int _stdcall HW_CC_Display(IN void * handle, void * hndd);
HWSDKSPIR_API int _stdcall HW_CC_SetImageModeNum(IN void * handle, unsigned int nNum);
HWSDKSPIR_API int _stdcall HW_CC_SetImageInfo(IN void * handle, IN OUT HW_IMAGE_BASIC_INFO * pInfo);
HWSDKSPIR_API int _stdcall HW_CC_SetDeviceInfo(IN void * handle, IN OUT HW_CC_DEVICE_INFO * pstDeviceInfo);
HWSDKSPIR_API int _stdcall HW_CC_SetBitMatchInfo(IN void * handle, IN OUT HW_BIT_MATCH_INFO * pInfo);
HWSDKSPIR_API int _stdcall HW_CC_RegisterImageBuffer(void * handle, unsigned char * pBuffer, unsigned int nBufferSize);

/* ===== */
/* 设置和获取相机参数的万能接口 */
/* ===== */

HWSDKSPIR_API int _stdcall HW_CC_GetInitValue(IN void * handle, IN const char * strValue, OUT HWSDK_INTVALUE * pInitValue);
HWSDKSPIR_API int _stdcall HW_CC_SetInitValue(IN void * handle, IN const char * strValue, IN OUT HWSDK_INTVALUE * pInitValue);

```

图2-25 引用命名空间

2.2.5 VS

使用工业相机 SDK 可开发 C++ 程序，在 SDK 开发目录下，提供 27 个 VS 示例程序，其中 6 个为 MFC 程序，21 个为控制台程序。本文档仅对 MFC 程序的开发步骤做简要介绍。



- C++示例程序兼容中英文，对关键的程序均有中英文的注释，界面控件均有中英文区分，可通过 MFC 属性界面的 Dialog 进行切换，跟随系统语言自动切换。
- 建议使用 VS2008 及以上版本。

前提条件

客户端安装成功的同时，32 位、64 位和 AnyCpu 的 dll 文件自动被写入 PC 环境变量。

操作步骤

1. 创建 VS 工程并添加引用, 加入 MvCameraControl.lib 和 MvCameraControl.h 到工程中, 如下图所示。

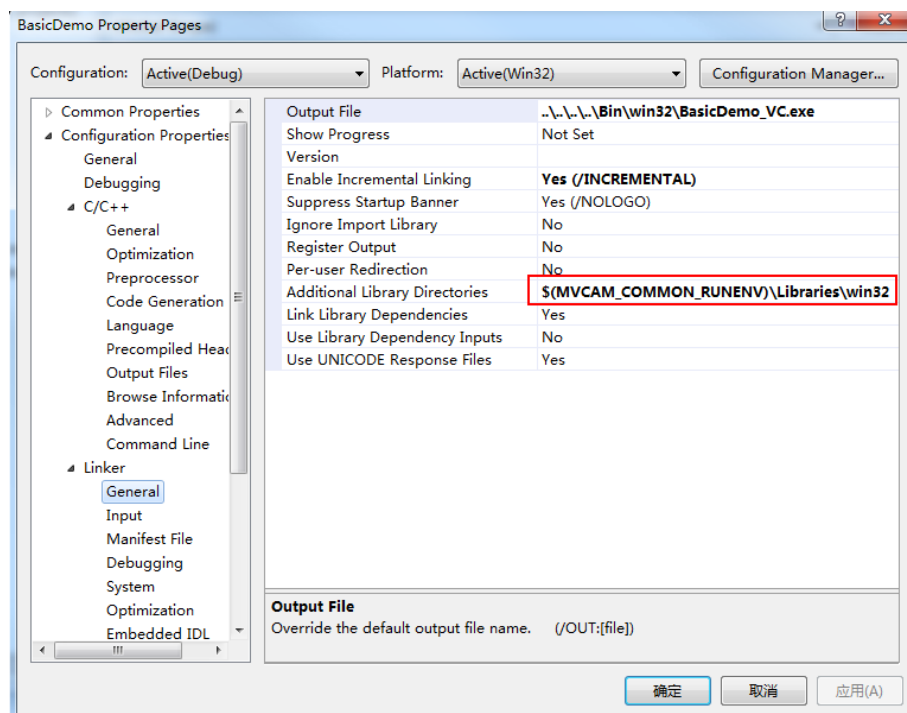


图2-26 加入 lib 库

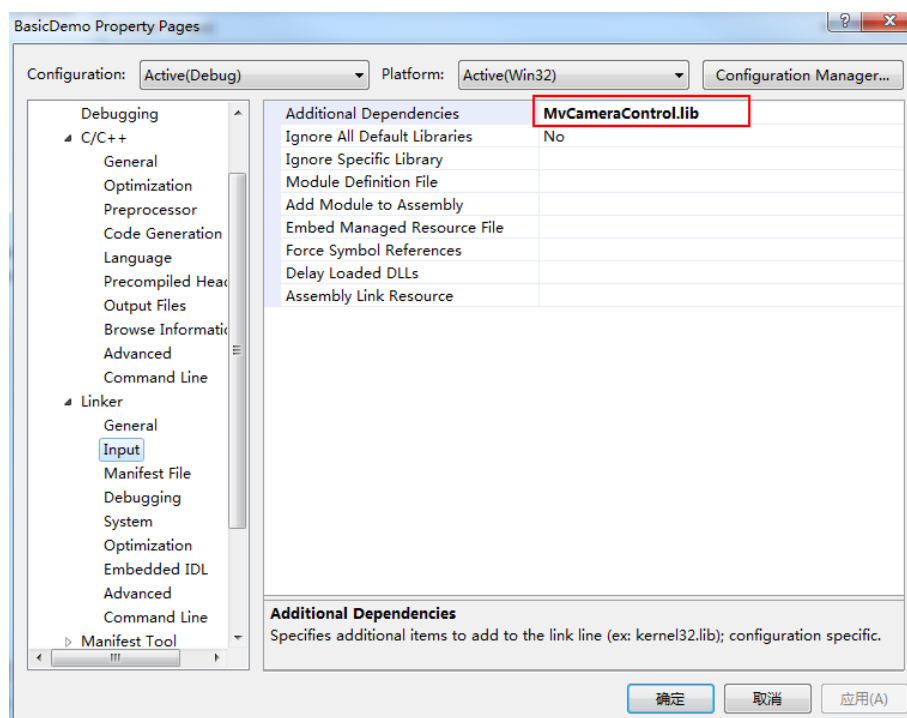


图2-27 加入 MvCameraControl.lib 文件

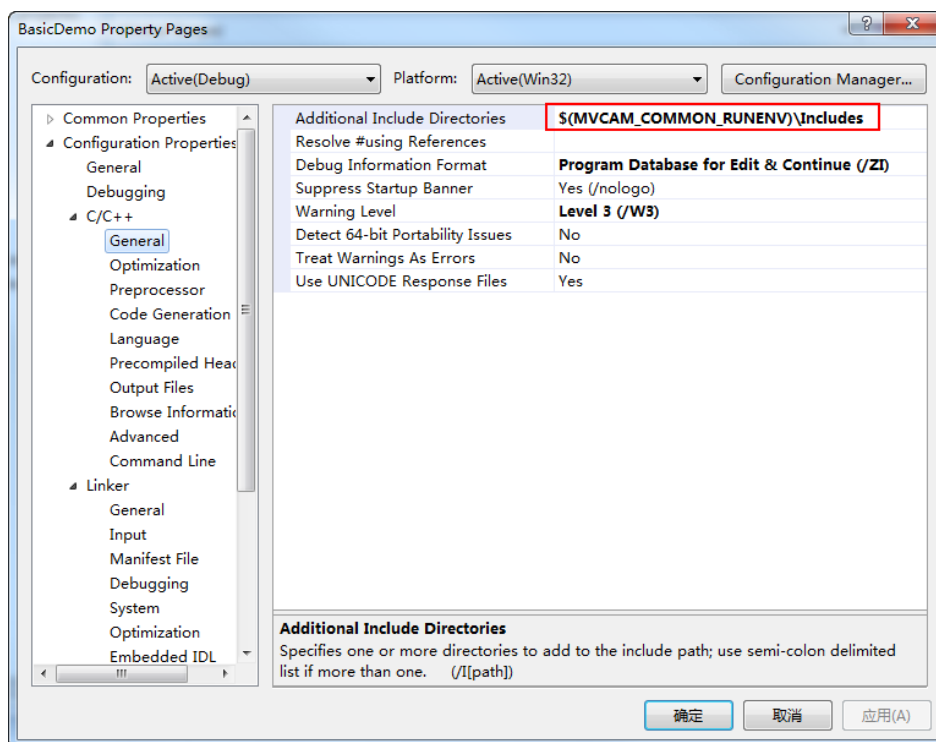


图2-28 加入 MvCameraControl.h 文件

2. 添加头文件和库文件引用，调用 MyCamera 中相机操作的函数，如下图所示。

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37
/* 以C++接口为基础，对常用函数进行二次封装，方便用户使用.....*/
/*.....*/
#ifdef MY_CAMERA_H
#define MY_CAMERA_H
#include <stdio.h>
#include "MvCameraControl.h"
class CMyCamera
{
public:
    CMyCamera();
    ~CMyCamera();
    static int EnumDevices(MV_CC_DEVICE_INFO_LIST* pstDevList);
    //ch:打开设备 | en:Open Device
    int Open(MV_CC_DEVICE_INFO* pstDeviceInfo);
    //ch:关闭设备 | en:Close Device
    int Close();
    //ch:开启抓图 | en:Start Grabbing
    int StartGrabbing();
    //ch:停止抓图 | en:Stop Grabbing
    int StopGrabbing();
    //ch:主动获取一帧图像数据 | en:Get one frame initiatively
    int GetOneFrameTimeout(unsigned char* pData, unsigned int* pnDataLen, unsigned int nDataSize, MV_FRAME_C
    //ch:设置显示窗口句柄 | en:Set Display Window Handle
    int Display(void* hWnd);
    //ch:保存图片 | en:save image
```

图2-29 引用命名空间

2.2.6 XE5

工业相机 SDK 可对接 XE5 软件，在 SDK 开发目录下，提供 1 个 MFC 示例程序。开发时需创建工程并配置头文件和 lib 目录，如下图所示。

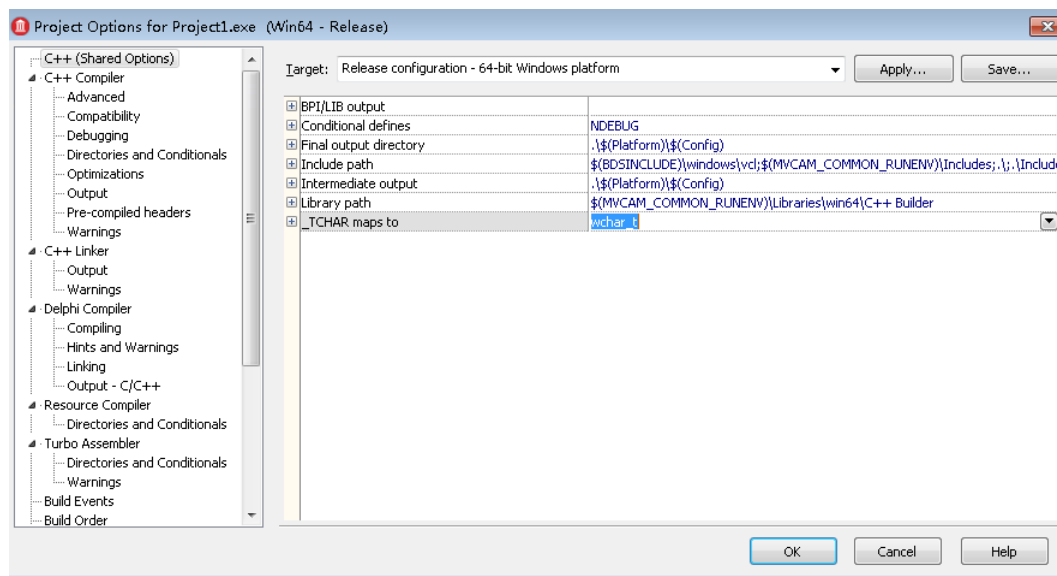


图2-30 创建工程并配置

说明

- MVS 安装成功的同时，32 位和 64 位的 dll 文件自动被写入 PC 环境变量中。
- XE5 Demo 与 C++程序链接使用的静态库不同，但函数接口一致，可调用相同头文件。

第3章 第三方 Demo 介绍

3.1 Halcon 示例程序

本节主要展示使用 Halcon 接口进行开发程序的方法及过程。在 SDK 开发包目录下，提供 5 个示例程序，请见下表。这些示例程序分别从不同角度展示利用 halcondontnet 和 MvCameraControl.Net 进行开发的方法。

表3-1 Halcon 示例程序

示例程序	开发语言	程序界面	功能说明
HalconGrabImage	C++	MFC 界面	基本示例程序，包含 Halcon 常用接口调用
Raw2Himage_C	C++	MFC 界面	使用 Halcon 接口进行图像像素转换以及显示示例程序
Raw2Himage_Csharp	C#	MFC 界面	
Raw_2_3DFile_C	C++	控制台界面	使用 Halcon 接口进行 3D 图像转换示例程序
Raw_2_3DFile_CSharp	C#	控制台界面	



- Raw2Himage_C 和 Raw2Himage_CSharp、Raw_2_3DFile_C 和 Raw_2_3DFile_CSharp 功能完全相同，只是开发的语言不同。
- C++与 C#版示例程序均兼容中英文，对关键的程序均有中英文的注释，界面控件均有中英文区分，可通过切换属性的 language 实现。
- 对于 C++示例程序，因 Halcon 版本差异，单个 Demo 无法适配所有的 Halcon 版本，故在工程名中备注适用的 Halcon 版本，如 HalconGrabImage_10 表示该示例用于 Halcon10，HalconGrabImage_11-13 表示该示例用于 Halcon11-13 版本。

3.1.2 使用步骤

软件

软件界面如下图所示，不同编译环境和示例程序下的软件界面略有差异，但功能基本一致，各区域功能介绍请见下表。

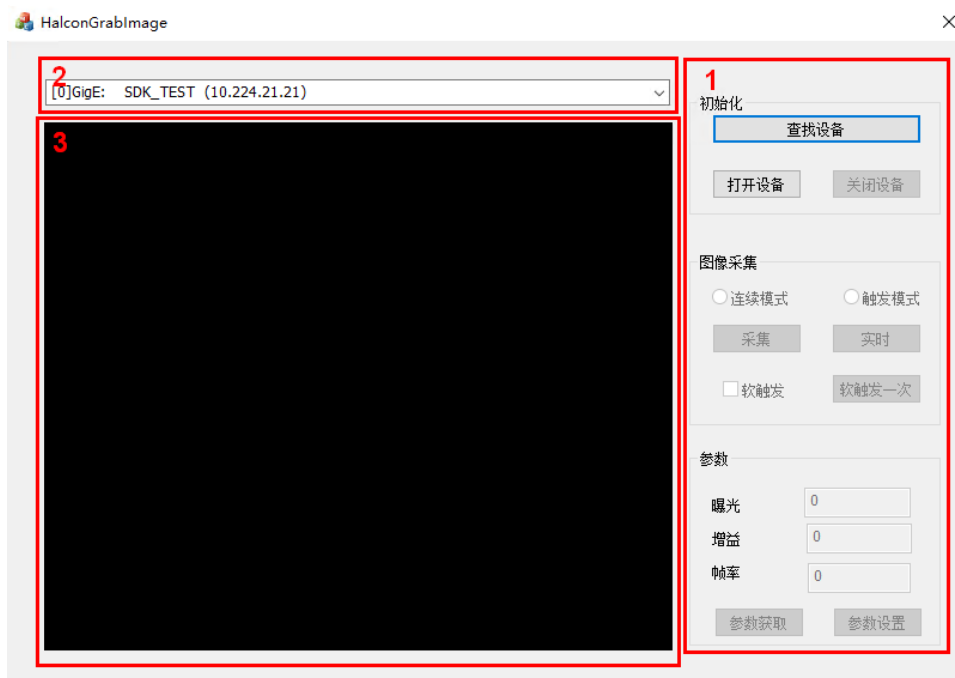


图3-1 Halcon 软件

表3-2 Halcon 软件功能说明

编号	名称	功能说明
1	控制模块	可进行初始化、图像采集、参数设置等操作。
2	下拉设备列表	下拉可显示设备进行连接。
3	图像显示区域	可显示已采集到的图像。

Halcon 软件基本操作步骤与 BasicDemo 相似，具体请见 2.1.1 BasicDemo 章节。

在使用 Halcon 软件前需拷贝 Halcon 插件。根据 Halcon 版本，在客户端安装目录下的 Development\ThirdPartyPlatformAdapter 路径中找到 hAcqMVision.dll 文件，将其拷贝到 Halcon 安装目录下的 HalconHDevelop 文件夹中。

说明

- 如果是 64 位，则拷贝到 64 位对应目录下。
- 若使用 Halcon XL 程序，则应拷贝 hAcqMVisionxl.dll，与 hAcqMVision.dll 使用方法相同。

控制台

控制台界面如下图所示，具有查找设备、打开设备、关闭设备、开始采集、停止采集、存取 Halcon3D 图像等功能。


```
d:\SVN\Release\MVS\Samples\Halcon\VC\Raw_2_3DFile_C\64\Debug\Raw_2_3DFile_C_11-13.exe
UserDefinedName: !@#$$#
[device 20]:
CurrentIp: 10.64.52.4
UserDefinedName: 600
[device 21]:
CurrentIp: 10.64.52.165
UserDefinedName: DSB123
[device 22]:
CurrentIp: 10.64.52.34
UserDefinedName: SmartCamera
[device 23]:
CurrentIp: 10.64.52.142
UserDefinedName: SmartCamera
[device 24]:
CurrentIp: 10.64.52.218
UserDefinedName: 2100Sub
Please Input camera index:10
Get One Frame: Width[1536], Height[1], FrameNum[1]
*****
* 0.ply; 1.obj; *
*****
Select FileType: 0
Press a key to exit.
```

图3-2 Halcon 控制台

操作步骤

1. 打开控制台界面后，Halcon 自动枚举在线设备；
2. 枚举完成后，输入对应枚举 3D 相机设备的下标，调用 SDK 中的取流函数，获取图像数据；
3. 根据提示，可选择将采集到的图像保存为 ply 或 obj 格式。

3.1.3 开发步骤

HalconGrabImage

HalconGrabImage 是一个基本示例程序，包含 Halcon 常用的接口调用，初次使用 Halcon 接口进行二次开发推荐首先参考 HalconGrabImage。

前提条件

MVS 客户端和 Halcon 插件安装成功的同时，32 位和 64 位的 dll 文件自动被写入 PC 环境变量中

操作步骤

1. 创建 C++工程并添加引用，如下图所示。

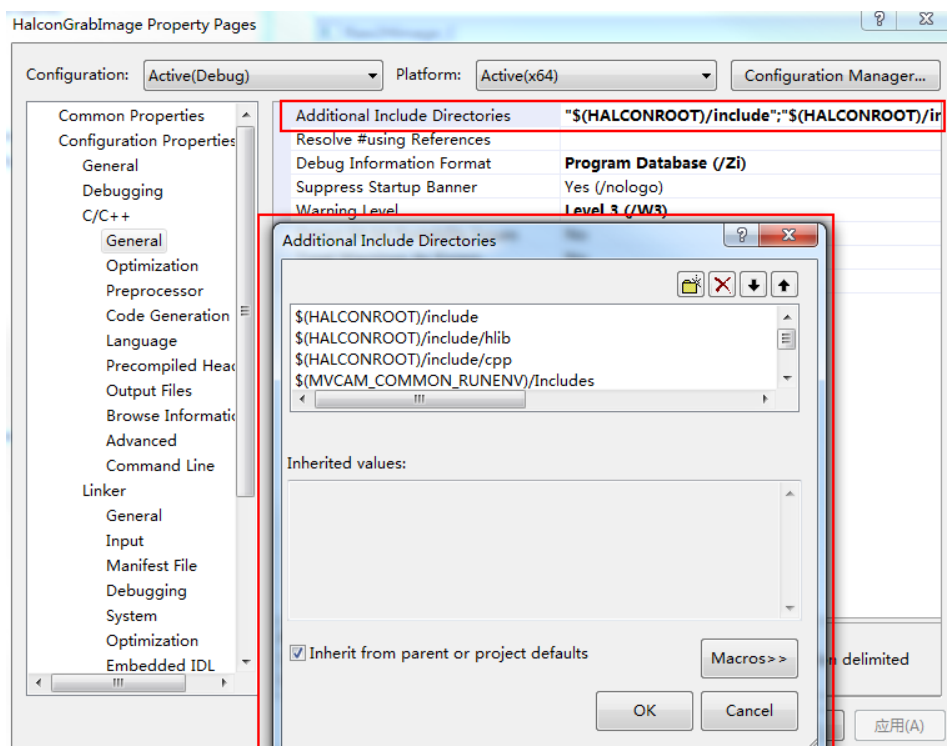


图3-3 创建工程

2. 将 Halcon 和 C++ 的 SDK 头文件和 lib 文件加入到工程中，如图 3-5 和图 3-4 所示。

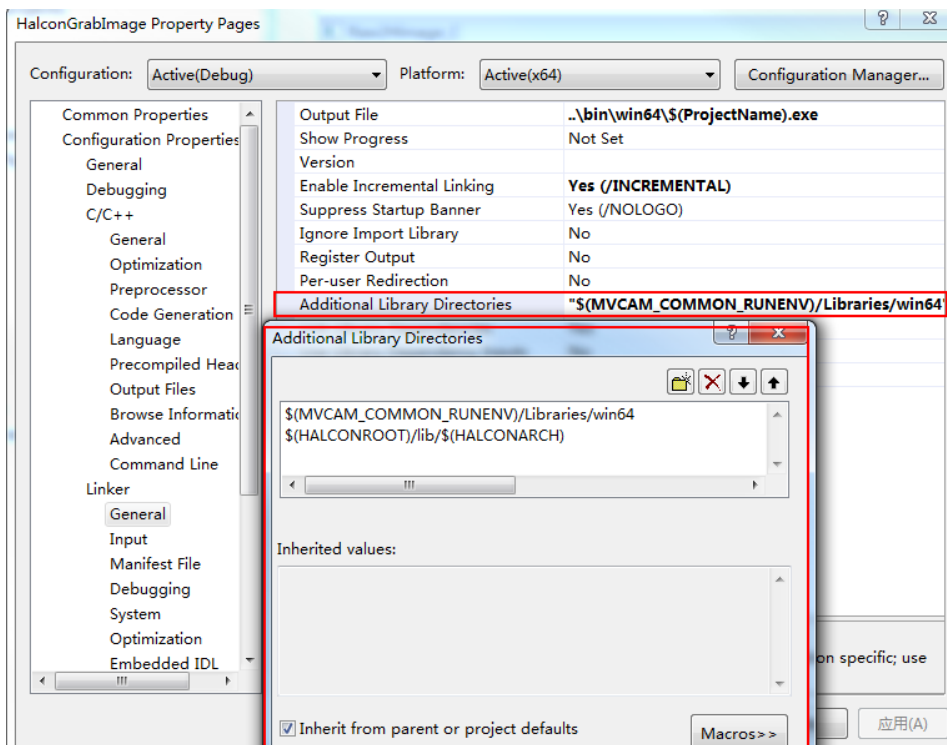


图3-4 加入 lib 库

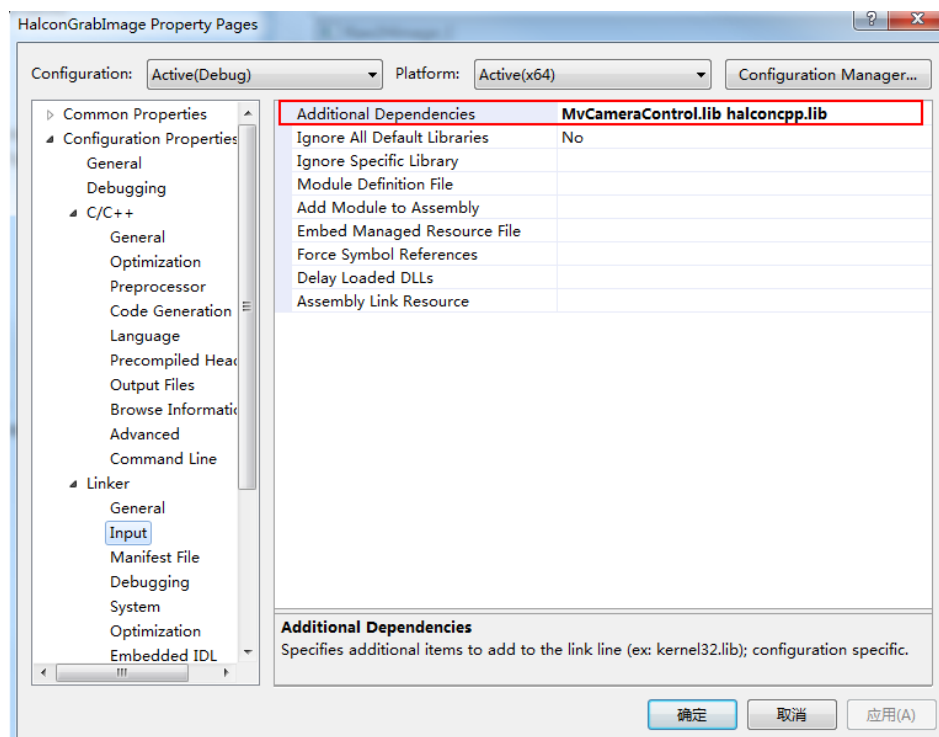


图3-5 加入 lib 文件

3. 添加引用后在工程中引用 MvCameraControl.h 和 HalconCpp.h，调用 Halcon 和 SDK 中相机操作的函数，如下图所示。

```
1 // HalconGrabImageDlg.h::头文件
2 //
3 //
4 #pragma once
5 #include "afxwin.h"
6 #include "MvCameraControl.h"
7 #include "HalconCpp.h"
8
9 using namespace Halcon;
10
11 /*函数返回码定义*/
12 typedef int Status;
13 #define STATUS_OK .....0
14 #define STATUS_ERROR .....-1
15
16 // HalconGrabImageDlg 对话框
17 class CHalconGrabImageDlg : public CDialog
18 {
19 public:
20 // 构造
21 CHalconGrabImageDlg(CWnd* pParent = NULL); // 标准构造函数
22
23 // 对话框数据
24 enum { IDD = IDD_HALCONGRABIMAGE_DIALOG };
25
26 protected:
27 virtual void DoDataExchange(CDataExchange* pDX); // DDX/DDV 支持
28
29 // 实现
30 protected:
31 HICON m_hIcon;
32
33 // 生成的消息映射函数
34 virtual BOOL OnInitDialog();
35 afx_msg void OnSysCommand(UINT nID, LPARAM lParam);
36
37 }
```

图3-6 引用头文件

Raw2Himage_C 和 Raw2Himage_Csharp

Raw2Himage_C 和 Raw2Himage_Csharp 展示如何使用 Halcon 接口进行图像像素转换以及显示。

前提条件

MVS 客户端和 Halcon 插件安装成功的同时，32 位和 64 位的 dll 文件自动被写入 PC 环境变量中。

操作步骤

1. 创建 C++工程并添加引用，如下图所示。

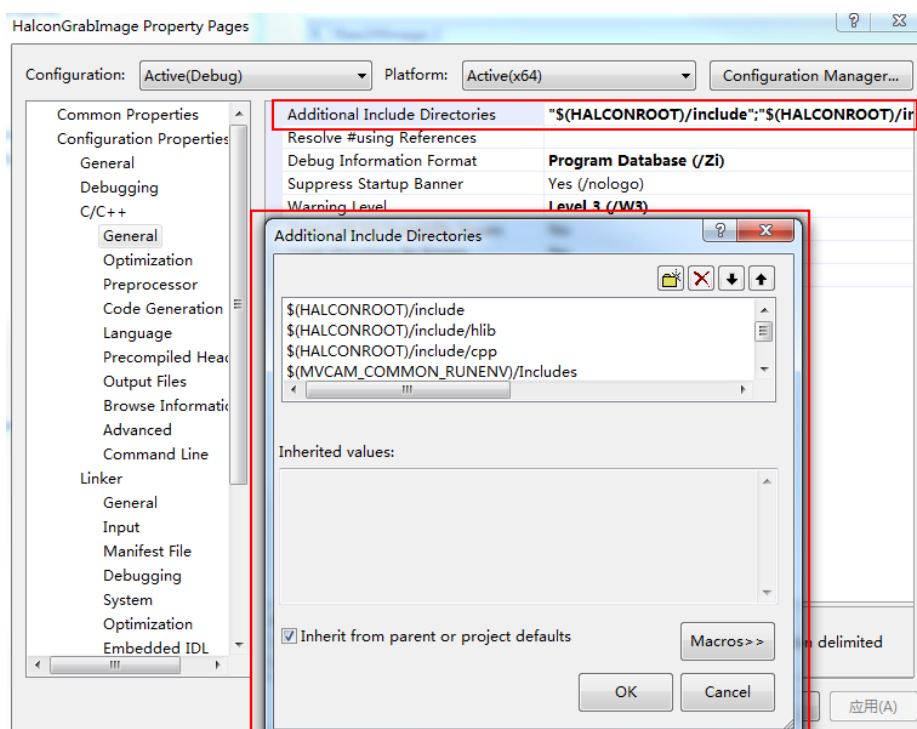


图3-7 创建工程

2. 将 Halcon 和 C++的 SDK 头文件和 lib 文件加入到工程中，如图 3-8 和图 3-9 所示。

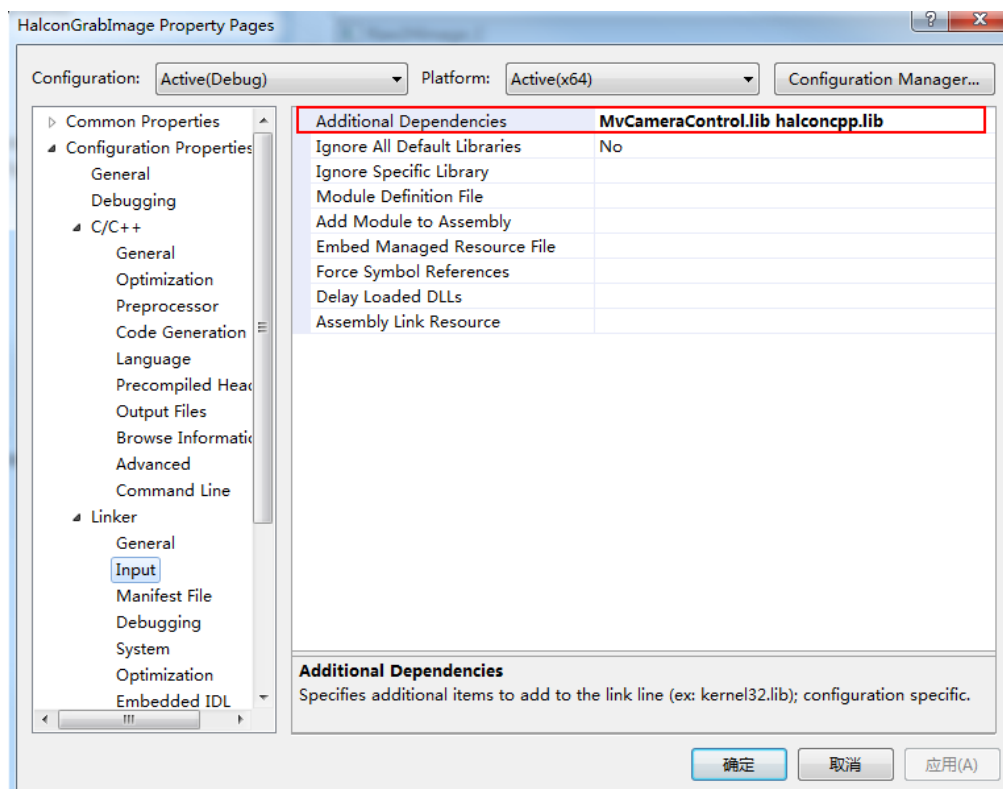


图3-8 加入 lib 文件

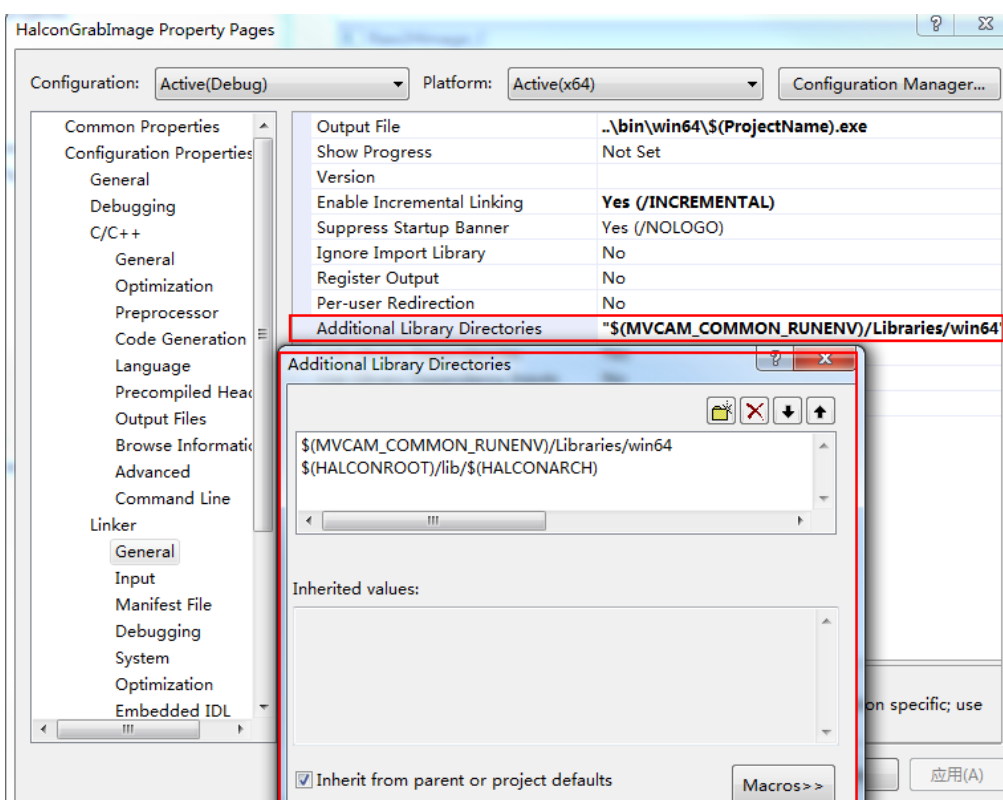


图3-9 加入头文件



说明

Raw2Himage_Csharp 中需创建 CS 工程并添加引用，将 halcondotnet.dll 和 MvCameraControl.Net.dll 文件加入到工程中，如下图所示。

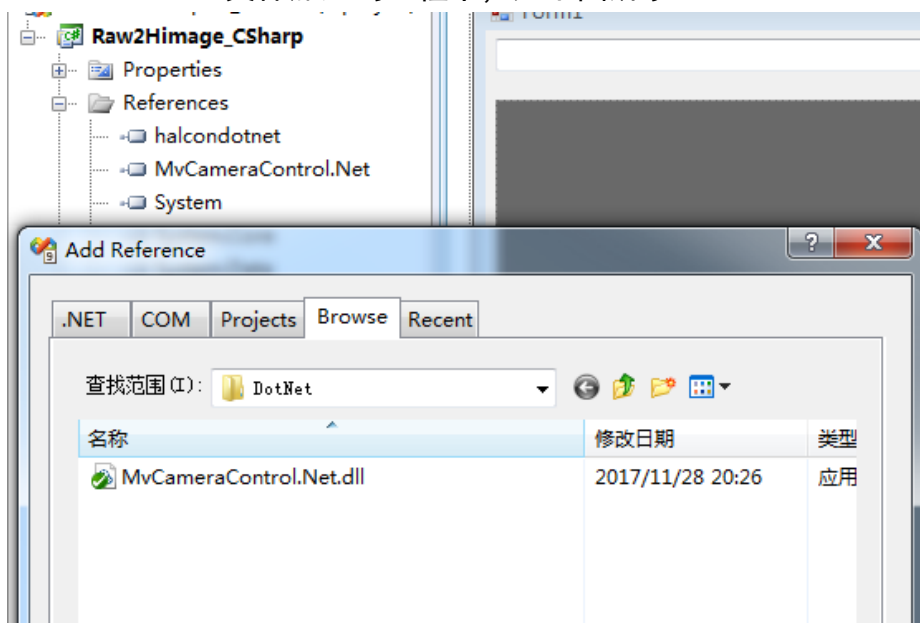


图3-10 Raw2Himage_Csharp 中加入 dll 文件

3. 添加引用后在工程中引用 MvCameraControl.h 和 HalconCpp.h，调用 Halcon 和 SDK 中相机操作的函数，如下图所示。

```

1
2 //HalconGrabImageDlg.h::头文件
3 //
4
5 #pragma once
6 #include "afxwin.h"
7 #include "MvCameraControl.h"
8 #include "HalconCpp.h"
9
10 using namespace Halcon;
11
12 /*函数返回码定义*/
13 typedef int Status;
14 #define STATUS_OK .....0
15 #define STATUS_ERROR .....-1
16
17 //CHalconGrabImageDlg 对话框
18 class CHalconGrabImageDlg : public CDialog
19 {
20 //构造
21 public:
22     CHalconGrabImageDlg(CWnd* pParent=NULL); //标准构造函数
23
24 //对话框数据
25     enum { IDD = IDD_HALCONGRABIMAGE_DIALOG };
26
27     protected:
28     virtual void DoDataExchange(CDataExchange* pDX); //DDX/DDV 支持
29
30 //实现
31 protected:
32     HICON m_hIcon;
33
34 //生成的消息映射函数
35     virtual BOOL OnInitDialog();
36     afx_msg void OnSvsCommand(UINT nID, LPARAM lParam);
37

```

图3-11 引用头文件



Raw2Himage_Csharp 中需引用命名空间 using MvCamCtrl.NET 和 using HalconDotNet, 如图 3-12 和图 3-13 所示。

```
.....public static object ByteToStruct(byte[] bytes, Type type);
.....public Ptr GetCameraHandle();
.....public int MV_CC_CloseDevice_NET();
.....public int MV_CC_ConvertPixelFormat_NET(ref MyCamera.MV_PIXEL_CONVERT_PARAM pstCvtParam);
.....public int MV_CC_CreateDevice_NET(ref MyCamera.MV_CC_DEVICE_INFO stDevInfo);
.....public int MV_CC_CreateDeviceWithoutLog_NET(ref MyCamera.MV_CC_DEVICE_INFO stDevInfo);
.....public int MV_CC_DestroyDevice_NET();
.....public int MV_CC_Display_NET(IntPtr hWnd);
.....public static int MV_CC_EnumDevices_NET(uint nLayerType, ref MyCamera.MV_CC_DEVICE_INFO_LIST stDevList);
.....public static int MV_CC_EnumDevicesEx_NET(uint nLayerType, ref MyCamera.MV_CC_DEVICE_INFO_LIST stDevList, string pM
.....public static int MV_CC_EnumerateTls_NET();
.....public int MV_CC_FeatureLoad_NET(string pFileName);
.....public int MV_CC_FeatureSave_NET(string pFileName);
.....public int MV_CC_FileAccessRead_NET(ref MyCamera.MV_CC_FILE_ACCESS pstFileAccess);
.....public int MV_CC_FileAccessWrite_NET(ref MyCamera.MV_CC_FILE_ACCESS pstFileAccess);
.....public int MV_CC_GetAcquisitionLineRate_NET(ref MyCamera.MVCC_INTVALUE pstValue);
.....public int MV_CC_GetAcquisitionMode_NET(ref MyCamera.MVCC_ENUMVALUE pstValue);
.....public int MV_CC_GetAllMatchInfo_NET(ref MyCamera.MV_ALL_MATCH_INFO pstInfo);
.....public int MV_CC_GetAOIOffsetX_NET(ref MyCamera.MVCC_INTVALUE pstValue);
.....public int MV_CC_GetAOIOffsetY_NET(ref MyCamera.MVCC_INTVALUE pstValue);
.....public int MV_CC_GetAutoExposureTimeLower_NET(ref MyCamera.MVCC_INTVALUE pstValue);
.....public int MV_CC_GetAutoExposureTimeUpper_NET(ref MyCamera.MVCC_INTVALUE pstValue);
.....public int MV_CC_GetBalanceRatioBlue_NET(ref MyCamera.MVCC_INTVALUE pstValue);
.....public int MV_CC_GetBalanceRatioGreen_NET(ref MyCamera.MVCC_INTVALUE pstValue);
.....public int MV_CC_GetBalanceRatioRed_NET(ref MyCamera.MVCC_INTVALUE pstValue);
.....public int MV_CC_GetBalanceWhiteAuto_NET(ref MyCamera.MVCC_ENUMVALUE pstValue);
.....public int MV_CC_GetBoolValue_NET(string strKey, ref bool pbValue);
.....public int MV_CC_GetBrightness_NET(ref MyCamera.MVCC_INTVALUE pstValue);
.....public int MV_CC_GetBurstFrameCount_NET(ref MyCamera.MVCC_INTVALUE pstValue);
.....public int MV_CC_GetDeviceInfo_NET(ref MyCamera.MV_CC_DEVICE_INFO pstDevInfo);
.....public int MV_CC_GetDeviceUserID_NET(ref MyCamera.MVCC_STRINGVALUE pstValue);
.....public int MV_CC_GetEnumValue_NET(string strKey, ref MyCamera.MVCC_ENUMVALUE pstValue);
.....public int MV_CC_GetExposureAutoMode_NET(ref MyCamera.MVCC_ENUMVALUE pstValue);
.....public int MV_CC_GetExposureTime_NET(ref MyCamera.MVCC_FLOATVALUE pstValue);
.....public int MV_CC_GetFloatValue_NET(string strKey, ref MyCamera.MVCC_FLOATVALUE pstValue);
.....public int MV_CC_GetFrameRate_NET(ref MyCamera.MVCC_FLOATVALUE pstValue);
```

图3-12 引用 using MvCamCtrl.NET 后调用接口

```
namespace HalconDotNet
{
    public class HOperatorSet
    {
        public HOperatorSet();

        public static void AbsDiffImage(HObject image1, HObject image2, out HObject imageAbsDiff, HTuple mult);
        public static void AbsFuncId(HTuple function, out HTuple functionAbsolute);
        public static void AbsImage(HObject image, out HObject imageAbs);
        public static void AbsInvarFourierCoeff(HTuple realInvar, HTuple imaginaryInvar, HTuple coefP, HTuple coefQ, HTu
        public static void AbsMatrix(HTuple matrixID, out HTuple matrixAbsID);
        public static void AbsMatrixMod(HTuple matrixID);
        public static void AccessChannel(HObject multiChannelImage, out HObject image, HTuple channel);
        public static void ActivateComputeDevice(HTuple deviceHandle);
        public static void AdaptTemplate(HObject image, HTuple templateID);
        public static void AddChannels(HObject regions, HObject image, out HObject grayRegions);
        public static void AddImage(HObject image1, HObject image2, out HObject imageResult, HTuple mult, HTuple add);
        public static void AddMatrix(HTuple matrixAID, HTuple matrixBID, out HTuple matrixSumID);
        public static void AddMatrixMod(HTuple matrixAID, HTuple matrixBID);
        public static void AddNoiseDistribution(HObject image, out HObject imageNoise, HTuple distribution);
        public static void AddNoiseWhite(HObject image, out HObject imageNoise, HTuple amp);
        public static void AddNoiseWhiteContourXld(HObject contours, out HObject noisyContours, HTuple numRegrPoints, HTu
        public static void AddSampleClassGmm(HTuple GMMHandle, HTuple features, HTuple classID, HTuple randomize);
        public static void AddSampleClassMlp(HTuple MLPHandle, HTuple features, HTuple target);
        public static void AddSampleClassSvm(HTuple SVMHandle, HTuple features, HTuple classVal);
        public static void AddSamplesImageClassGmm(HObject image, HObject classRegions, HTuple GMMHandle, HTuple randomiz
        public static void AddSamplesImageClassMlp(HObject image, HObject classRegions, HTuple MLPHandle);
        public static void AddSamplesImageClassSvm(HObject image, HObject classRegions, HTuple SVMHandle);
        public static void AdjustMosaicImages(HObject images, out HObject correctedImages, HTuple from, HTuple to, HTuple
        public static void AffineTransContourXld(HObject contours, out HObject contoursAffinTrans, HTuple homMat2D);
        public static void AffineTransImage(HObject image, out HObject imageAffinTrans, HTuple homMat2D, HTuple interpola
        public static void AffineTransImageSize(HObject image, out HObject imageAffinTrans, HTuple homMat2D, HTuple inter
        public static void AffineTransObjectModel3d(HTuple objectModel3DID, HTuple homMat3D, out HTuple objectModel3DIDaf
        public static void AffineTransPixel(HTuple homMat2D, HTuple row, HTuple col, out HTuple rowTrans, out HTuple colT
        public static void AffineTransPoint2d(HTuple homMat2D, HTuple px, HTuple py, out HTuple qx, out HTuple qy);
```

图3-13 引用 using HalconDotNet 后调用接口

- 调用 SDK 中的开始取流接口 StartGrabbing, 并创建一个线程 WorkThread, 在该线程中循环调用 SDK 的获取一帧图像的接口 GetOneFrameTimeout。



Raw2Himage_Csharp 中需调用开始取流接口 MV_CC_StartGrabbing_NET，并创建 ReceiveImageWorkThread 线程，在该线程中循环获取一帧图像接口 MV_CC_GetOneFrameTimeout_NET。

5. 获取图像后进行图像格式转换，保存成 Himage 图片格式，并用 Halcon 自带的显示接口 HalconDisplay 进行显示。

Ram_2_3DFile_C 和 Raw_2_3DFile_Csharp

Raw_2_3DFile_C 和 Raw_2_3DFile_CSharp 重点展示如何使用 Halcon 接口进行 3D 图像的转换。

相机基本操作流程与 HalconGrabImage 相似。本节重点介绍如何将相机的 3D 数据转换为 Halcon 中 3D 的数据格式，并存取为文件，存取的文件类型有 ply 和 obj 两种。

操作步骤

1. 调用 SDK 中的开始取流接口 StartGrabbing；
2. 调用取流接口 MV_CC_GetOneFrameTimeout 获取一帧图像；
3. 获取图像后进行图像格式转换，保存为 ply 和 obj 的 3D 数据格式。

3.2 Labview 示例程序

本节主要展示使用工业相机 SDK 开发 Labview 程序的方法及过程。在 SDK 开发包目录下，提供两个 Labview 的 MFC 示例程序，分别为单相机示例程序 Samples 和双相机示例程序 TwoCameraSamples。

3.2.1 Labview VI

为方便用户在 Labview 平台调用 C 接口的 SDK，对 C 接口 SDK 进行 Labview VI 封装，封装成支持 Labview 调用的子 VI。在 MvCameraLib.lvlib 中，一共有 39 个子 VI，不同子 VI 的功能如下表所示。

表3-3 Labview VI

子 VI	功能说明	子 VI	功能说明
EnumDevices.vi	枚举设备	SetIntValue.vi	设置整型节点属性值
EnumDevicesEx.vi	枚举指定厂商的设备	GetEnumValue.vi	获取枚举型节点属性值
IsDeviceAccessible.vi	判断设备是否可达	SetEnumValue.vi	设置枚举型节点属性值
IsGigEDeviceAccessible.vi	判断 GigE 设备是否可达	GetBoolValue.vi	获取布尔型节点属性值

工业相机 SDK 示例程序和插件使用说明

IsU3VDeviceAccessible.vi	判断 U3V 设备是否可达	SetBoolValue.vi	设置布尔型节点属性值
CreateHandle.vi	创建设备句柄	GetFloatValue.vi	获取浮点型节点属性值
CreateHandleWithoutLog.vi	创建设备句柄，不生成日志	SetFloatValue.vi	设置浮点型节点属性值
DestroyHandle.vi	销毁句柄	GetStringValue.vi	获取字符串型节点属性值
OpenDevice.vi	打开设备	SetStringValue.vi	设置字符串型节点属性值
CloseDevice.vi	关闭设备	SetCommandValue.vi	设置命令型节点属性值
GetDeviceInfo.vi	获取设备信息	FileAccessRead.vi	从设备读取文件
GetAllMatchInfo.vi	获取各种类型的信息	FileAccessWrite.vi	将文件写入设备
GetGigEMulticastStatus.vi	获取 GigE 设备的组播状态	SaveImageEx2.vi	保存图片，支持 Bmp 和 Jpeg
StartGrabbing.vi	开始取流	SaveImageToFile.vi	保存图像到文件
StopGrabbing.vi	停止取流	SavePointCloudData.vi	保存 3D 点云数据，支持 PLY、CSV 和 OBJ 三种格式
GetOneFrame.vi	获取一帧图像(主动取图)	StartRecord.vi	开始录像
GetOneFrameTimeout.vi	获取一帧图像(采用超时机取图)	StopRecord.vi	停止录像
ConvertPixelFormat.vi	像素格式转换	InputOneFrame.vi	输入一帧录像数据
DisplayOneFrame.vi	显示一帧图像	GigEIssueActionCommand.vi	发出 GigE 设备动作命令
GetIntValue.vi	获取整型节点属性值		

以上 VI 是对 C 接口 SDK 的二次封装，其内部调用 C 接口 SDK 中的一些动态链接库。

以 EnumDevices.vi 为例，接口的输入输出参数如下表所示。

表3-4 EnumDevices.vi 输入输出参数

参数	参数类型	参数释义
nLayerType	输入参数	32 位整型数据，表示传入枚举的设备类型，1 为 GigE 相机，4 为 U3V 相机

错误输入	输入参数	用户可自定义。若设置错误输入，当子 VI 运行失败时，返回错误输出；若不设置，当子 VI 运行失败时，不返回错误输出
function return	输出参数	接口调用返回值，返回 0 表示调用成功，返回负数表示调用失败的错误码
DeviceNum	输出参数	枚举到的相机数目
pstGigEDevArray	输出参数	返回 GigE 相机设备信息簇数组，设备信息簇定义如图 3-14 所示
pstU3VDevArray	开始取流	返回 U3V 相机设备信息簇数组
错误输出	输出参数	当子 VI 运行失败时，可返回相应错误码

```
typedef struct _MV_CC_DEVICE_INFO_
{
    ....unsigned short.....nMajorVer;.....
    ....unsigned short.....nMinorVer;.....
    ....unsigned int.....nMacAddrHigh;...
    ....unsigned int.....nMacAddrLow;...
    ....unsigned int.....nTLayerType;...

    ....unsigned int.....nReserved[4];...

    ....union
    ....{
    ....MV_GIGE_DEVICE_INFO.stGigEInfo;.....
    ....MV_USB3_DEVICE_INFO.stUsb3VInfo;.....
    ....//more....
    ....}SpecialInfo;

}MV_CC_DEVICE_INFO;
```

图3-14 设备信息簇定义

3.2.2 使用步骤

Labview Demo 界面如下图所示，各区域功能介绍请见下表。

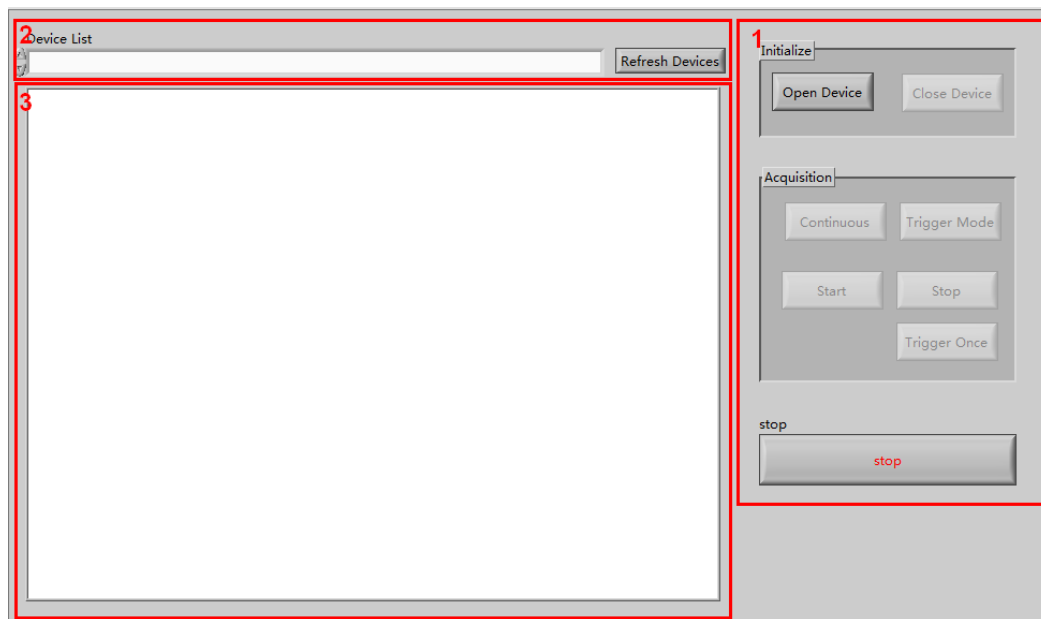


图3-15 Labview Demo

表3-5 Labview Demo 功能说明

编号	名称	功能说明
1	控制模块	可进行刷新设备、初始化、图像采集等操作。
2	下拉设备列表	下拉可显示设备进行连接。
3	图像显示区域	可显示已采集到的图像。

说明

TwoCameraSamples 示例程序的软件界面中有 2 个图像显示区域，其余各区域功能和操作步骤均与 CameraSample 示例程序一致。

操作步骤

1. 点击“Refresh Devices”查找设备，此时下拉设备列表区域出现当前在线的设备列表。
2. 选择其中一个设备，点击“Open Device”打开当前选中的设备，打开设备后默认以连续方式取流，如下图所示。

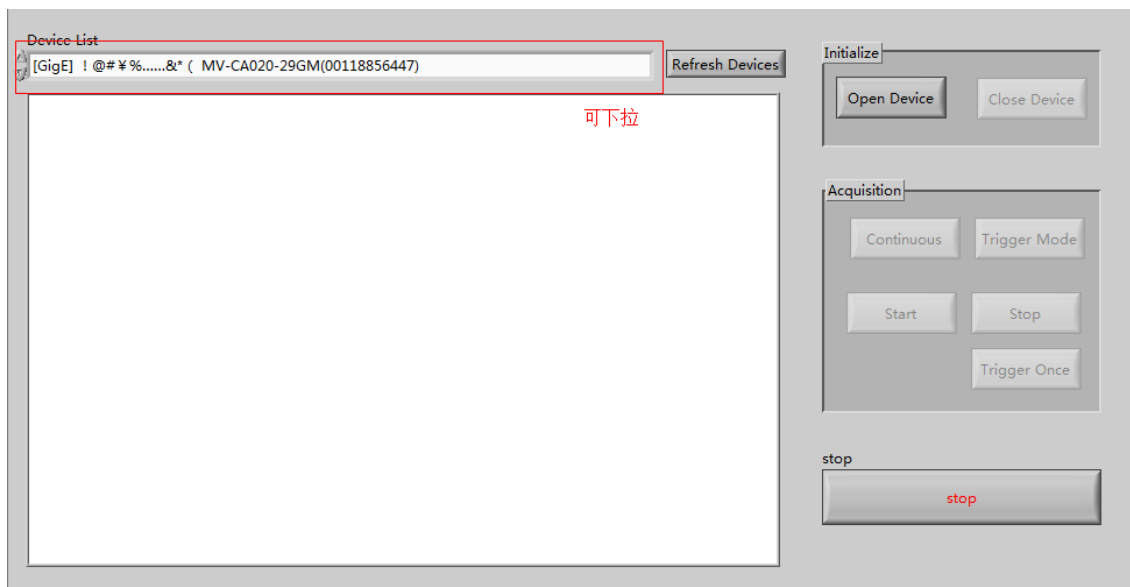


图3-16 打开设备

3. 通过控制模块下的“Acquisition”，可选择连续模式“Continuous”或触发模式“Trigger Mode”。
- 选择触发模式“Trigger Mode”时，点击“Start”后，可通过点击“Trigger Once”完成触发一次功能，如下图所示。

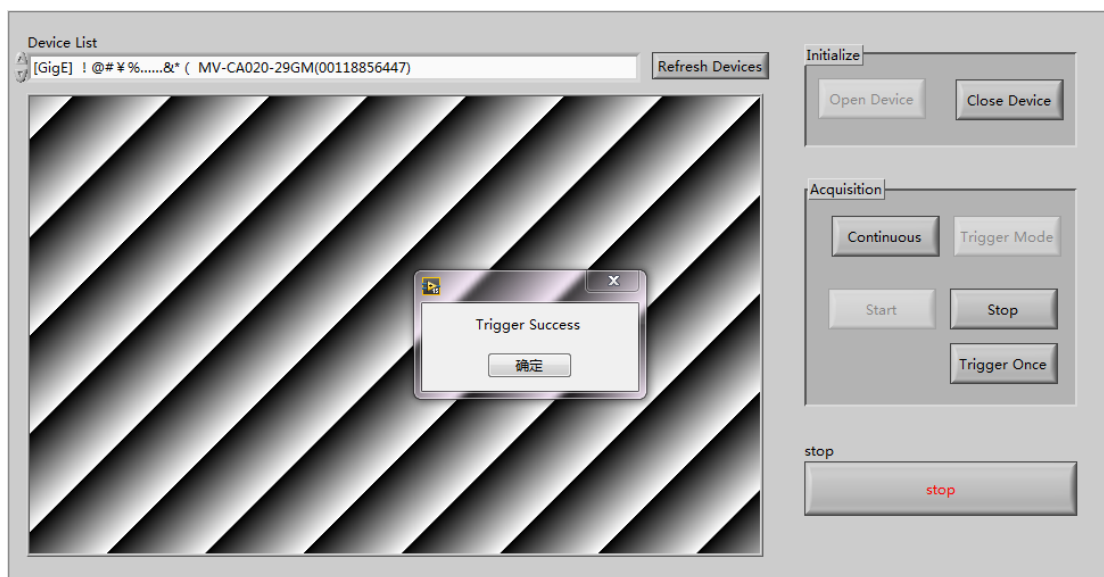


图3-17 触发成功

- 选择连续模式“Continuous”时，点击“Start”进行图像采集，图像显示区域出现实时图像，如下图所示。

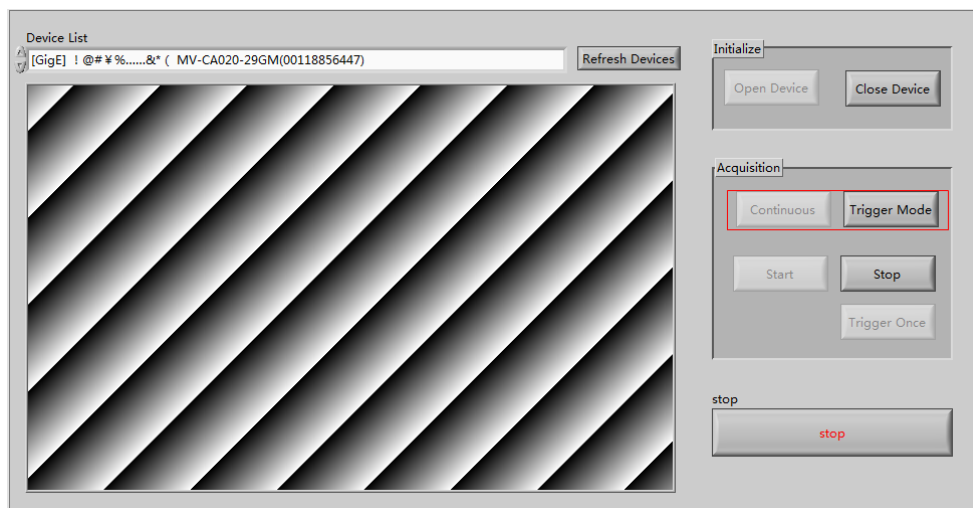


图3-18 开始采集

3.2.3 开发步骤

单相机示例程序 Samples 和双相机示例程序 TwoCameraSamples 的开发步骤一致。

操作步骤

1. 在项目管理器中右击我的电脑，选择添加文件，将 MvCameraLib.lvlib 加入项目中。
2. 新建一个 VI，例如命名为 DisplayPanel.vi 或 TwoCameraDisplayPanel.vi，如下图所示。

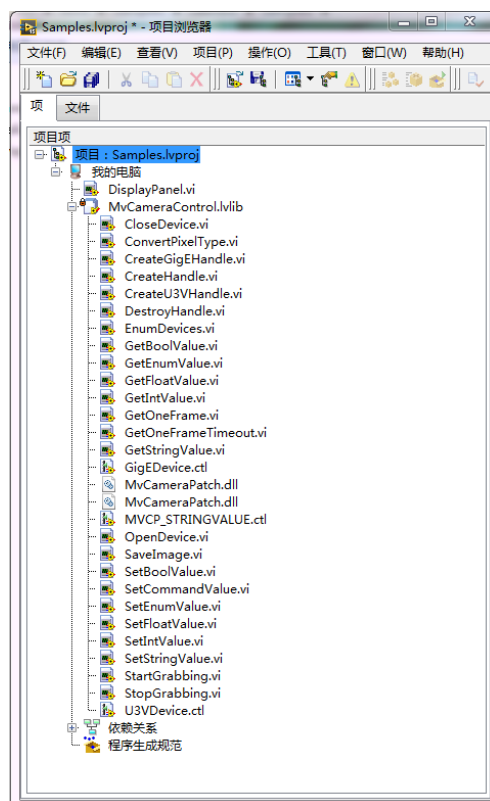


图3-19 新建项目

3. 在 VI 的程序框图面板添加 MvCameraLib.lvlib 中的子 VI，即可进行相机操作。



Labview 在加载 VI 初始化过程中，会出现如下图所示界面。该界面会自动搜索 VI 调用到的 MvCameraControl.dll 和 MvCameraPatch.dll。如果无法搜索到，手动点击“浏览”，添加 SDK 中的 MvCameraControl.dll 和 MvCameraPatch.dll 文件路径即可。

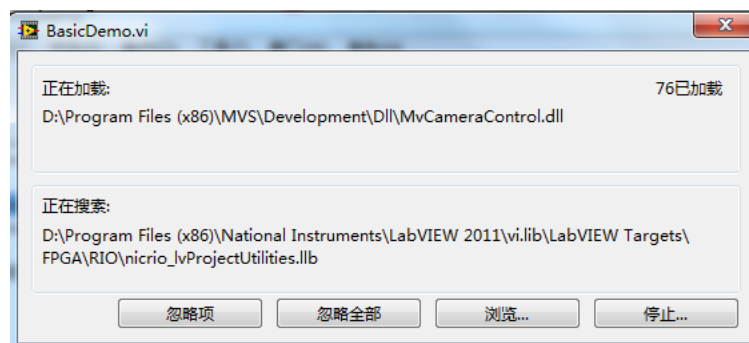


图3-20 加载 VI 初始化

第4章 插件介绍

4.1 DirectShow

本章节主要介绍 DirectShow 连接相机时的插件使用，通过本套插件可连接工业相机。

4.1.1 环境配置

相机环境配置

在使用 DirectShow 插件前，需要对相机环境进行配置。

操作步骤

1. 打开客户端，若 PC 与相机不在同一网段，则配置 IP，配置方式请参考客户端用户手册。
2. 设置相机参数，确保相机能在客户端上正常取流。

插件的注册与注销



在客户端安装完成后，默认不注册该插件，需要用户手动注册、注销与检验。注册和注销脚本放在客户端安装目录下 Development\ThirdPartyPlatformAdapter\DirectShow 文件夹，分 x64 和 x86 两个版本，这里以 64 位版本为例。



图4-1 注册和注销脚本

MvDSS2 目录存放最新版插件的注册脚本，MvDSS 目录存放之前版本的插件注册脚本，推荐使用新版本插件，不同版本插件请见下表。

表4-1 DriectShow 插件

插件	版本	说明
register.bat	旧版本	注册脚本
unregister.bat		卸载脚本
InstallDSSvc_x64.bat	新版本	注册脚本  说明 x86 版本为 InstallDSSvc.bat
UnInstallDSSvc_x64.bat		卸载脚本  说明 x86 版本为 UnInstallDSSvc.bat



说明

脚本需要以管理员权限运行。

运行注册脚本后，可用 graphedt.exe 程序查看是否注册成功，如下图所示。

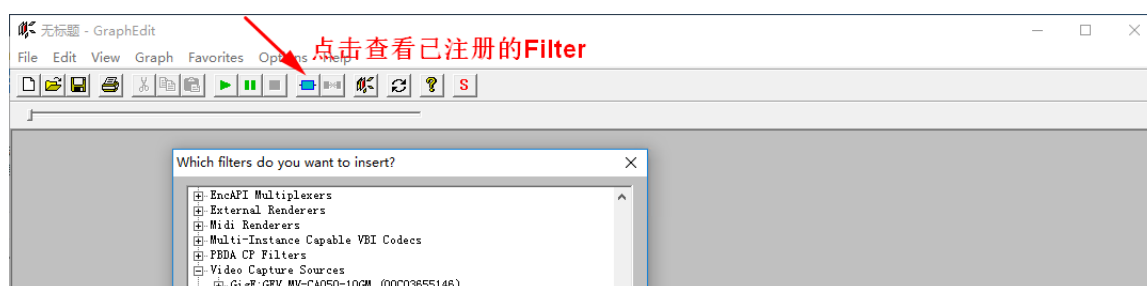


图4-2 注册成功结果

每个相机作为子项注册在 Video Capture Source 项内，该列表每 3 秒钟刷新一次。

4.1.2 使用说明

采集显示图像

在 Direct 插件中，可使用 phedt.exe 采集并显示图像的具体。

操作步骤

1. 在 graphedt.exe 程序中打开菜单 Graph-Insert Filters;
2. 在 Video Capture Source 项内选择已注册的相机项，双击将相机添加到面板中;

3. 在相机 Filter 上右键点击 Filter Properties 查看相机 Filter 信息，包括两个标签页，CameraInfo 显示相机信息和图像配置信息，ParameterTree 显示相机属性树；
4. 在菜单栏选择 Graph->Insert Filters，在 DirectShow Filters 中选择 Video Renderer，并将相机 Filter 的 Video Out 针脚与 Video Renderer 的 Input 针脚连接，DirectShow 会自动加载合适的 Filter 对相机 Filter 进行渲染，如下图所示；

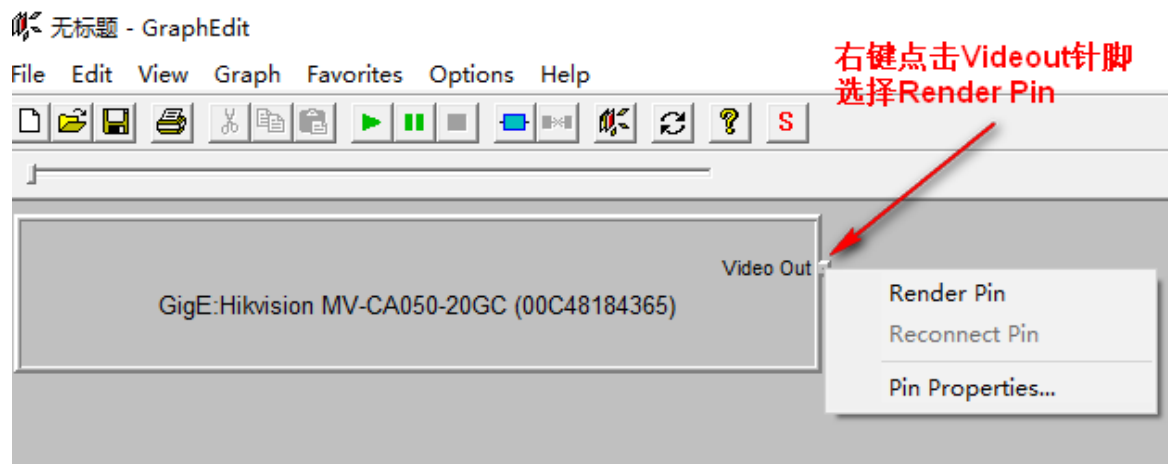


图4-3 添加渲染 Filter

5. 点击菜单的 Graph-Play 即可进行图像预览。

修改相机参数

Filter 属性页包含 CameraInfo 标签和 ParameterTree 标签，如图 4-4 和图 4-5 所示。CameraInfo 标签页显示相机信息和图像配置信息，ParameterTree 标签页显示相机支持的所有属性，可在该界面查看和修改相机属性。

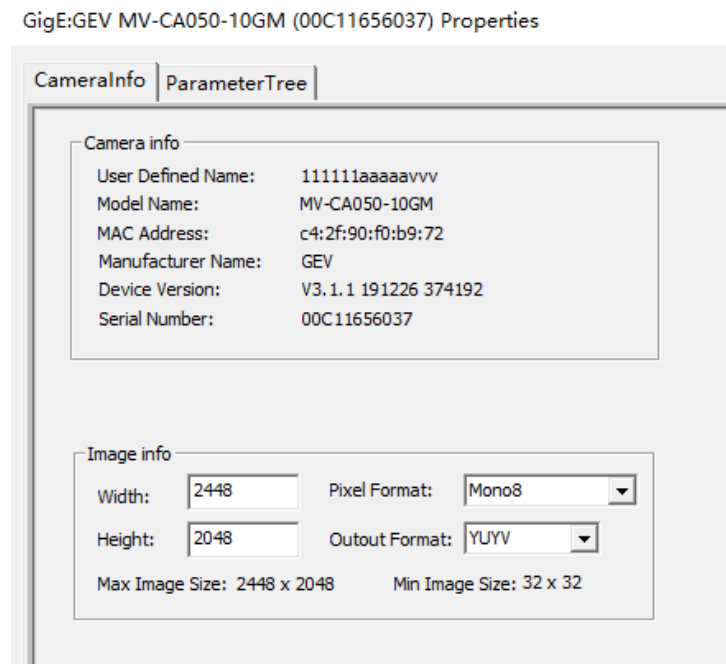


图4-4 相机信息页面

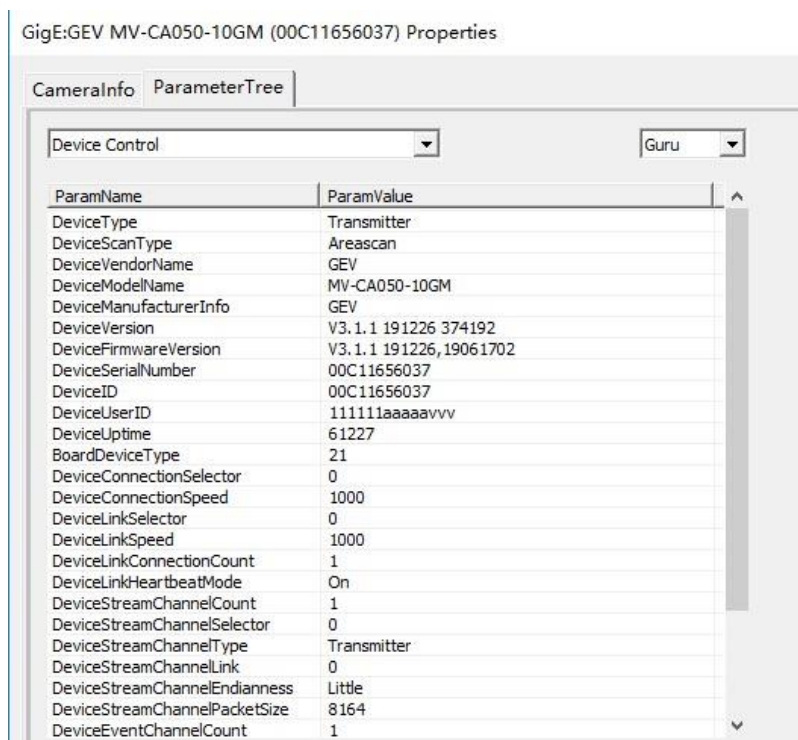


图4-5 相机属性树页面

在相机属性树页面，可根据属性目录和用户等级筛选要显示的属性，用户等级从低到高依次为 Beginner、Expert 和 Guru，用户等级越高，显示的属性越全面。

若要对某个相机属性进行修改，可双击 ParamValue 一列的值，进入可编辑状态，修改参数值。枚举类型的属性以下拉框形式进行编辑，命令类型的属性会显示执行按钮，其他类型属性以字符串形式进行编辑。

4.1.3 示例程序

DirectShowDisplay MFC 示例程序展示通过 DirectShow Filter 进行枚举相机、打开关闭相机、相机取流、参数获取与设置等基本操作，供用户参考，路径为 Development\Samples\DirectShow\DirectShowDisplay（适用于新插件）。MvDSSource2.h 头文件中包含相机列表获取、参数获取、参数设置等接口可供外部调用。此示例程序可编译 x86 和 x64 版本，分别对应插件的 x86 和 x64 版本。

4.2 Halcon

本节主要介绍 Halcon 连接相机时插件的使用，通过本套插件可连接工业相机。



说明

- 该插件接口支持相机的所有节点。
- 在客户端安装路径下的 Halcon 目录中提供 Halcon 接口取流 Demo，该 Demo 为软件界面程序，软件界面如下图所示。



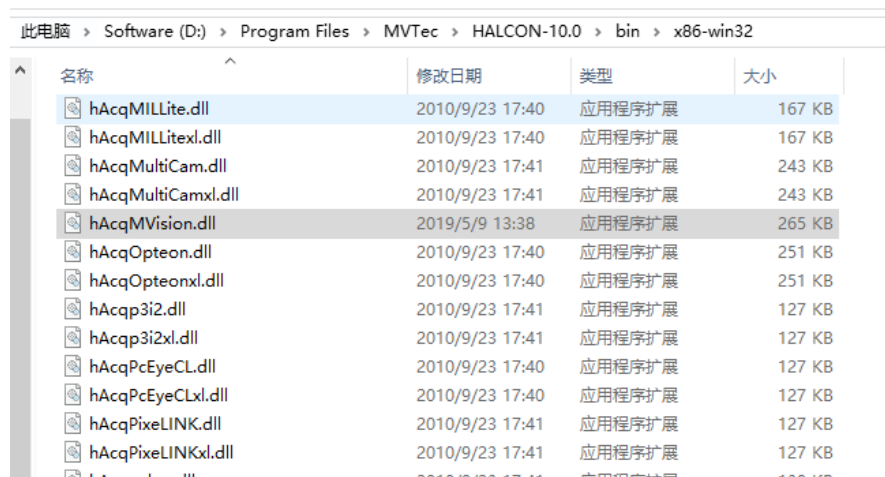
图4-6 Halcon 接口取流 Demo

4.2.2 环境配置

使用 Halcon 插件前需进行环境配置。

操作步骤

1. 根据 Halcon 版本，在客户端安装目录下的 Development\ThirdPartyPlatformAdapter 路径中找到 hAcqMVision.dll 文件，将其拷贝到 Halcon 安装目录下的 HalconHDevelop 文件夹中，如下图所示。



名称	修改日期	类型	大小
hAcqMILLite.dll	2010/9/23 17:40	应用程序扩展	167 KB
hAcqMILLitexl.dll	2010/9/23 17:40	应用程序扩展	167 KB
hAcqMultiCam.dll	2010/9/23 17:41	应用程序扩展	243 KB
hAcqMultiCamxl.dll	2010/9/23 17:41	应用程序扩展	243 KB
hAcqMVision.dll	2019/5/9 13:38	应用程序扩展	265 KB
hAcqOpteon.dll	2010/9/23 17:40	应用程序扩展	251 KB
hAcqOpteonxl.dll	2010/9/23 17:40	应用程序扩展	251 KB
hAcqp3i2.dll	2010/9/23 17:41	应用程序扩展	127 KB
hAcqp3i2xl.dll	2010/9/23 17:41	应用程序扩展	127 KB
hAcqPcEyeCL.dll	2010/9/23 17:40	应用程序扩展	127 KB
hAcqPcEyeCLxl.dll	2010/9/23 17:40	应用程序扩展	127 KB
hAcqPixelINK.dll	2010/9/23 17:41	应用程序扩展	127 KB
hAcqPixelINKxl.dll	2010/9/23 17:41	应用程序扩展	127 KB

图4-7 拷贝动态库



说明

- 如果是 64 位，需拷贝到 64 位对应目录下。
 - 若使用 Halcon XL 程序，则应拷贝 hAcqMVisionxl.dll，与 hAcqMVision.dll 使用方法相同。
 - 该 dll 文件是可以重命名的，但是一定要保留“hAcq”的前缀。例如重命名为 hAcqImageAcquistion.dll 时，在“图像获取接口”的下拉框中选择名字为 ImageAcquistion 的驱动。
2. 打开客户端，若 PC 与相机不在同一网段，则配置 IP，配置方式请参考客户端用户手册。
 3. 设置相机参数，确保相机能在 MVS 上正常取流。

4.2.3 相机配置

通过 Halcon 插件可进行简单相机配置。

驱动选择配置

操作步骤

1. 打开 Halcon 菜单栏的助手，打开新的 Image Acquisition。
2. 在弹出的 Image Acquistion 对话框中，选中“图像获取接口”，在下拉框中选择“MVision”，如下图所示。



图4-8 驱动选择配置

设备选择配置

操作步骤

1. 在 Image Acquisition 对话框中切换到“连接”的界面。
2. 在设备列表中选择想要连接的设备，点击“连接”即可，如下图所示。

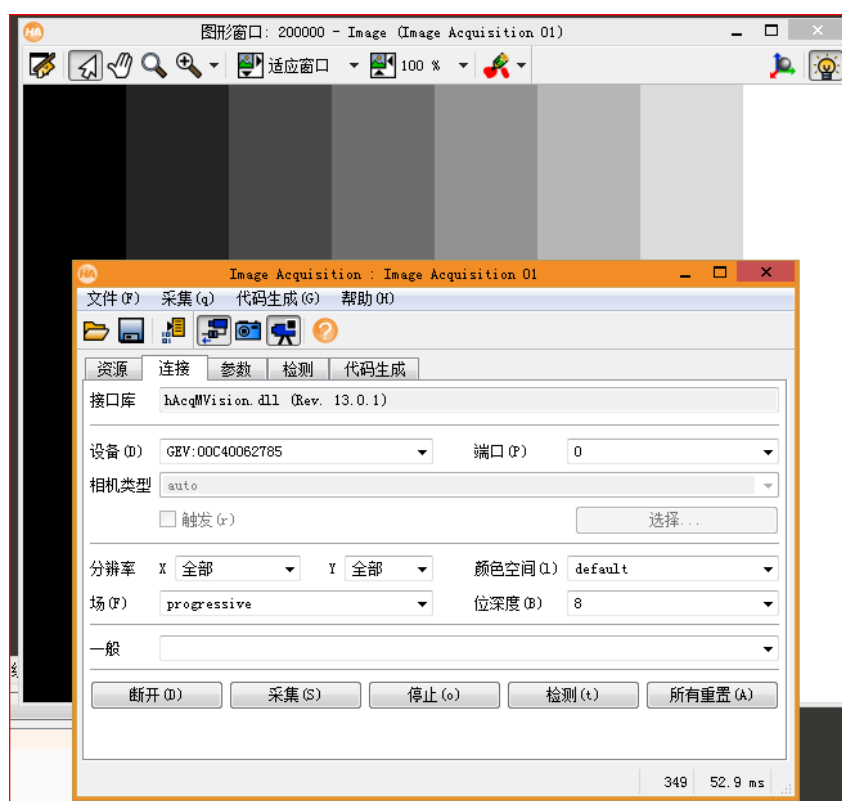


图4-9 设备选择配置

设备操作

点击 Image Acquisition 对话框的“实时”按钮，进行实时画面的播放，可在“图形窗口”看到画面，点击“停止”可以停止实时播放。如果没有弹出“图形窗口”的对话框，可在菜单栏中“可视化>打开图形窗口”中打开图像显示对话框。

参数配置

在 Image Acquisition 对话框中可以切换到“参数”界面，在此界面可以对相机的常用属性参数进行设置，如下图所示。

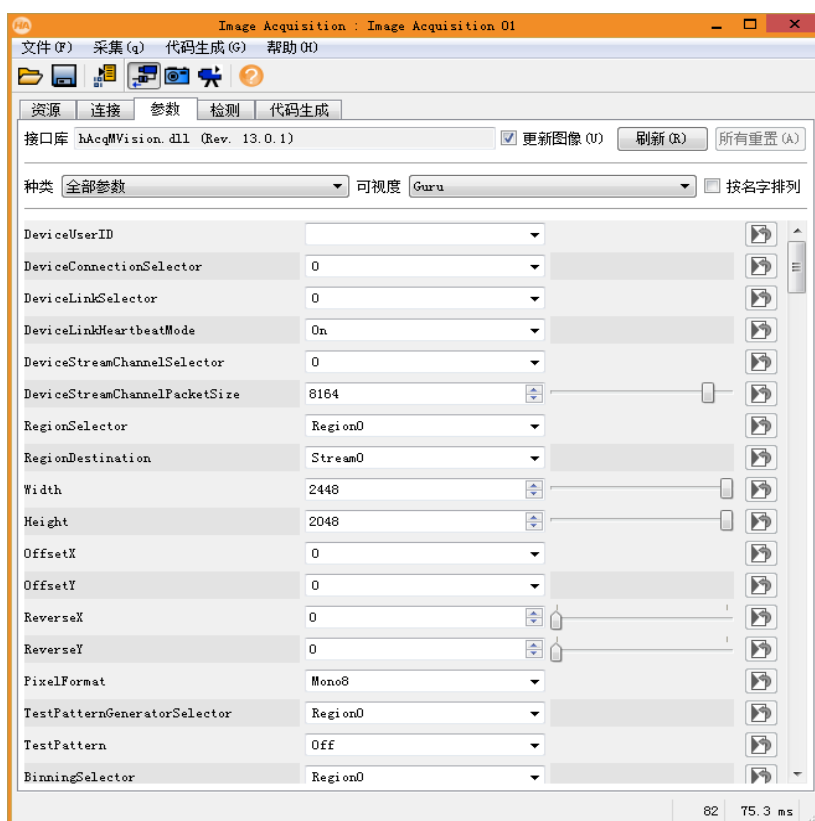


图4-10 参数配置



说明

若找不到某些参数设置，可查看用户等级是否为大师等级，并刷新参数列表。

4.3 Sherlock

本节主要介绍 Sherlock 连接相机时的插件使用，通过本套插件可连接工业相机。

4.3.1 环境配置

在使用 Sherlock 插件前需进行环境配置。

操作步骤

1. 根据 Sherlock 版本，在客户端安装目录下的 Development\ThirdPartyPlatformAdapter 路径中找到 UsrAcqDrv.dll 文件,将其拷贝到 Sherlock 安装目录下的 Drivers 文件夹中，如下图所示。

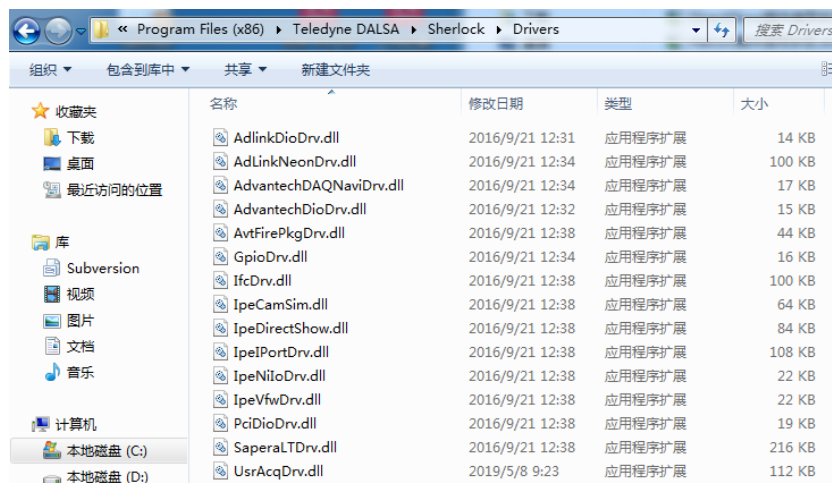


图4-11 拷贝动态库



说明

如果是 64 位，需拷贝到 64 位对应目录下。

2. 打开客户端，若 PC 与相机不在同一网段，则配置 IP，配置方式请参考客户端用户手册。
3. 设置相机参数，确保相机能在客户端上正常取流。
4. 打开 Sherlock 菜单栏中的选项>采集>Sample driver，设置为 Enabled，如下图所示。

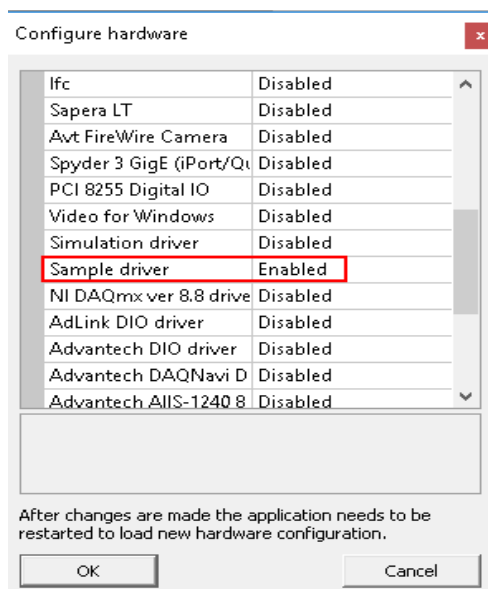


图4-12 相机驱动配置

5. 关闭 Sherlock，重启生效。

4.3.2 相机配置

通过 Sherlock 插件可进行简单相机配置。

开启触发模式

操作步骤

1. 打开 Sherlock 菜单栏中的图像>选项，通过勾选“External Camera Trigger”设置触发模式 Line0，若不勾选则为连续模式。
2. 调整“Non-triggered acquisition timeout”处的数值，设置连续模式超时时间。
3. 勾选“On acquisition timeout: Skip execution of this ‘image window’ and continue executing program without a timeout error”。

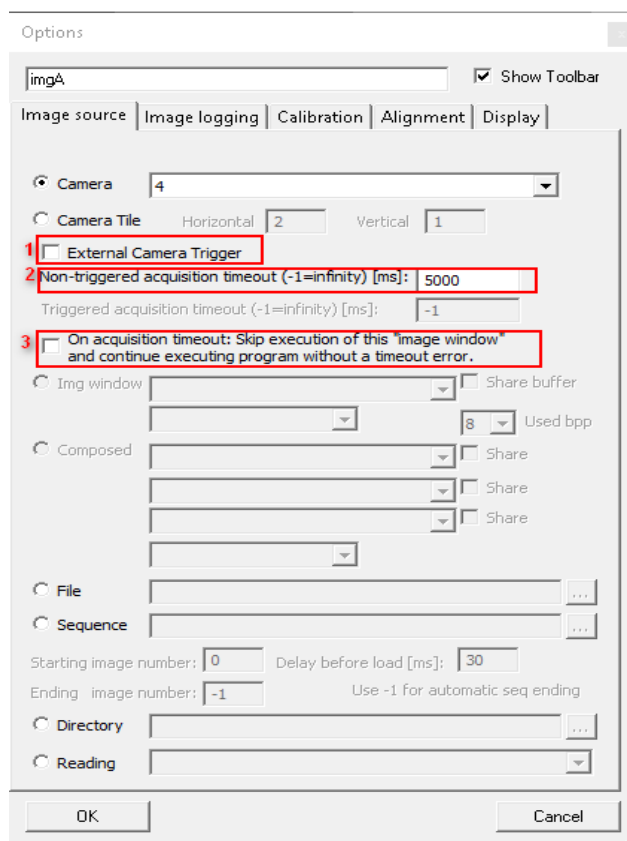


图4-13 开启触发模式

IO 输出

打开 Sherlock 菜单栏中的 view，勾选 Digital Outputs 使 sherlock 的右下角显示相机的 IO 输出情况。

IO 输出为 on 时，相机输出 IO 信号，此时对应相机勾选 line inverter 参数；IO 输出为 off 时，相机不输出 IO 信号，此时对应相机不勾选 line inverter 参数。

Sherlock 内部开启多个相机

操作步骤

1. 打开 Sherlock 菜单栏中的图像>新建，开启新窗口，如下图所示。

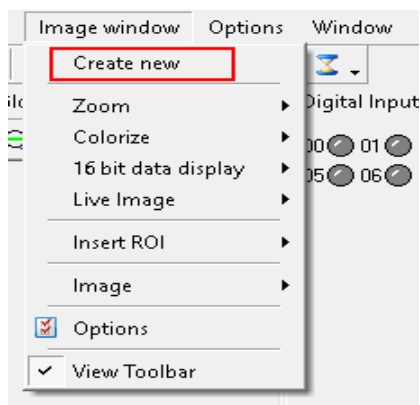


图4-14 开启新窗口

2. 右键新窗口图像，选择 Camera，即可选择新窗口所需的取流相机，如下图所示。

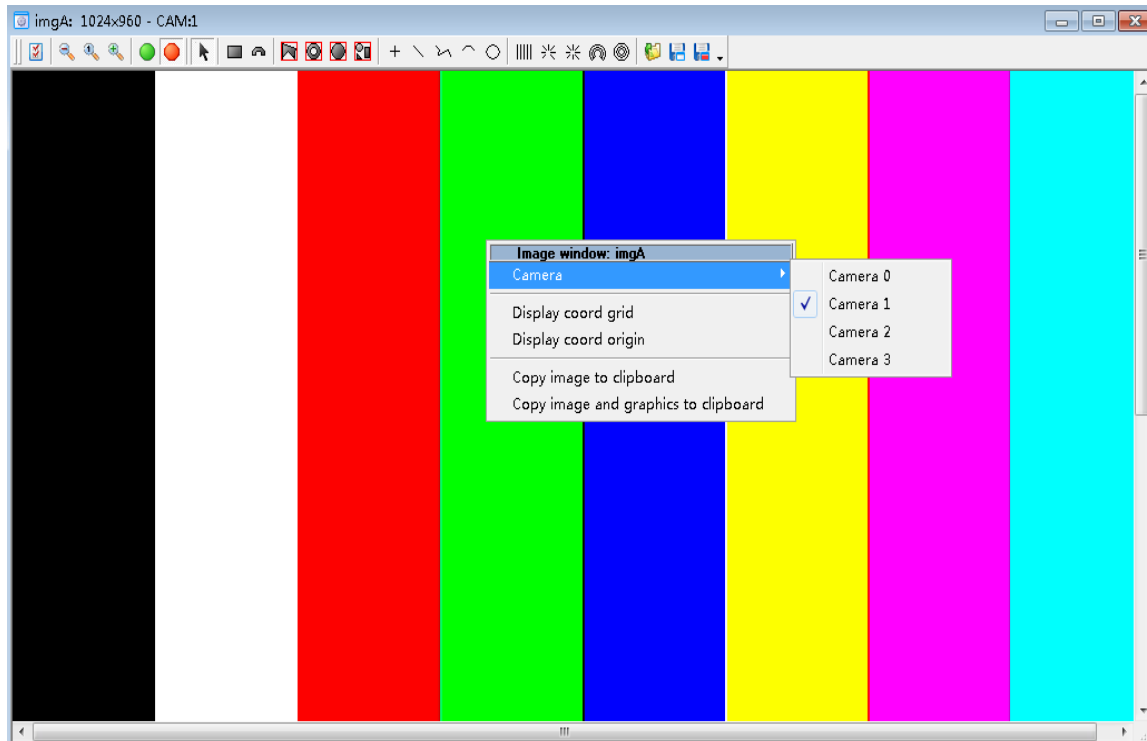


图4-15 选择相机

4.3.3 其他功能

打开多个 Sherlock

操作步骤

1. 在路径 C:\ProgramData\Teledyne DALSA\Sherlock\Drivers 下找到 Drivers.ini，打开 Drivers.ini 文件，将文件中 Sample driver 的 ENABLE 设置成 1，如下图所示，保存关闭。

```

NAME      = Simulation driver
FILE      = IpeCamSim.dll
CONFIG    = IpeCamSim.ini
ENABLED   =0

[DRIVER7]
NAME      = Sample driver
FILE      = UstrAcqDrv.dll
CONFIG    = UstrAcqDrv.ini
ENABLED   =1

[DRIVER8]
NAME      = NI DAQmx ver 8.8 driver
FILE      = IpeNiIoDrv.dll
CONFIG    = IpeNiIoDrv.ini
ENABLED   =0

```

图4-16 设置 Drivers.ini 文件



ProgramData 文件夹默认被隐藏，需要将隐藏文件夹显示出来才能找到相应路径。

2. 复制两份 Drivers.ini 文件，放在 Drivers.ini 同一目录下，分别命名为 Driver1.ini 和 Drivers2.ini，并编辑文件中的 Sample driver，如下图所示，注意驱动中的 ENABLE 设置成 1，保存关闭。

<pre> [DRIVER7] NAME = Sample driver FILE = UstrAcqDrv.dll CONFIG = UstrAcqDrv1.ini ENABLED =1 </pre>	<pre> [DRIVER7] NAME = Sample driver FILE = UstrAcqDrv.dll CONFIG = UstrAcqDrv2.ini ENABLED =1 </pre>
--	--

图4-17 复制编辑 2 份 Drivers.ini 文件

3. 在 C:\Program Files (x86)\Teledyne DALSA\Sherlock\Bin 或者 C:\ProgramData\Teledyne DALSA\Sherlock\Drivers 路径下新建两个文件：UstrAcqDrv1.ini 和 UstrAcqDrv2.ini。此时这两个文件名对应步骤 2 中 Drivers1.ini 和 Drivers2.ini 中的 CONFIG。
4. 填入需要打开相机的序列号或者相机的用户自定义名称，如图 4-18 和图 4-19 所示。

如果相机的序列号和用户自定义名称都存在，默认只加载序列号。单个 ini 文件最多可填入 16 个相机，填入完成后保存关闭。

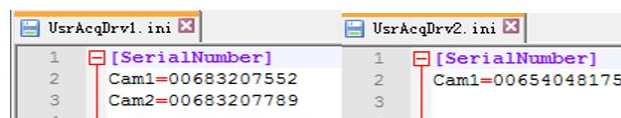


图4-18 通过序列号打开相机

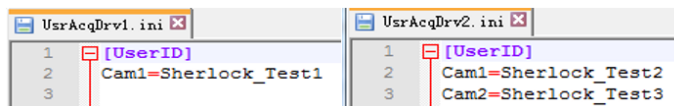


图4-19 通过用户自定义名称打开相机

5. 打开 Sherlock 软件，选择主菜单 Program->Save As，设置文件名为 MyTest.ivs，保存在路径 C:\Program Files (x86)\Teledyne DALSA\Sherlock\Bin 下，关闭 Sherlock 软件。
6. 在 C:\Program Files (x86)\Teledyne DALSA\Sherlock\Bin 路径下新建文件：Cam1.bat 和 Cam2.bat。
 - 将 Cam1.bat 编辑为：start "" "C:\Program Files (x86)\Teledyne DALSA\Sherlock\Bin\IpeStudio.exe" -acq:Drivers1.ini -load: MyTest.ivs;
 - 将 Cam2.bat 编辑为：start "" "C:\Program Files (x86)\Teledyne DALSA\Sherlock\Bin\IpeStudio.exe" -acq:Drivers2.ini -load: MyTest.ivs。

说明

- 这里的 C:\Program Files (x86)\Teledyne DALSA\Sherlock\Bin 是 win7 32bit 环境下 sherlock 的安装路径，如果是 win7 64bit，路径应该为 C:\Program Files (x86)\Teledyne DALSA\Sherlockx64\Bin。
 - ini 和 ivs 文件名称都要步骤 3 和步骤 5 中的命名对应。
7. 双击 Cam1.bat，可以打开一个 sherlock 连接相机 1 和相机 2；双击 Cam2.bat，打开第二个 sherlock 连接相机 1。如果想要改变连接的相机，只需修改 UsrAcqDrv*.ini 中的序列号即可。

说明

UsrAcqDrv*.ini 文件如果不存在或者 UsrAcqDrv*.ini 文件中的关键字 Cam*不存在，则会默认枚举到所有相机。

GenTL 相机连接

Sherlock 支持通过修改配置文件 UsrAcqDrv.ini 连接 GenTL 相机，修改格式如下图所示。

```
[GenTLPath]
PATH=C:\Program Files (x86)\Common Files\MVS\Runtime\Win64_x64\MvProducerVIR.cti

[InterfaceID]
IF1=Virtual GigE Vision

[SerialNumber]
Cam1=Vir54260486
```

图4-20 修改配置文件

- GenTLPath: 指定 cti 文件的路径。
- InterfaceID: 要打开的 interface 编号，最多支持 8 个 interface，从 IF1 到 IF8，FG0 为采集卡的 interface 名。
- SerialNumber: 需要连接的相机序列号，最多支持 16 个，从 Cam1 到 Cam16。

UsrAcqDrv.ini 文件配置

由于 Sherlock 软件没有选择指定相机并打开的功能，因此选择并打开相机的操作需要放到插件内部去实现。但插件内部没有可视化界面，所以需要创建一个配置文件 UsrAcqDrv.ini，通过读配置文件中相机的信息来打开指定的相机并取流。

不同节点的配置内容如图 4-21 和表 4-2 所示，节点类型对照表请见表 4-3。

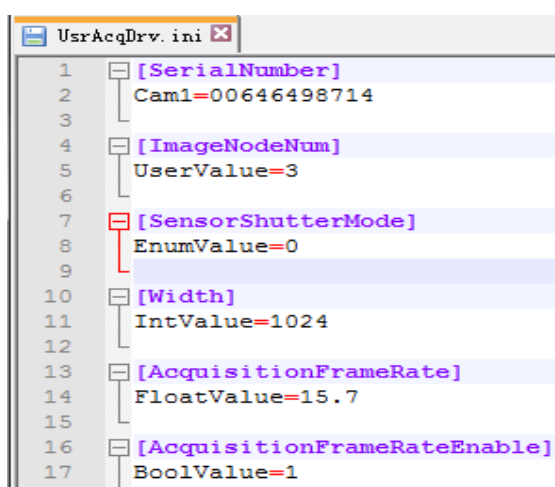


图4-21 配置文件格式

表4-2 配置文件内容

配置节点	配置内容
SerialNumber	节点配置要打开的相机（也可以使用 UserID 节点），单个 ini 文件最多配置 16 个相机，即从 Cam1 到 Cam16。
ImageNodeNum	节点配置图片缓存节点。 格式为 UserValue=value，UserValue 代表此节点为用户需求节点，value 为要设置的值。



说明
其余属性配置请参考客户端中的属性树。

表4-3 节点类型对照表

ini 配置文件中的节点类型	MVS 属性树中的节点类型
UserValue	代表此节点为用户自定义节点
EnumValue	IEnumeration
IntValue	IInteger
FloatValue	IFloat
BoolValue	IBoolean
StringValue	IString

IO:Camera 支持

用户可通过 IO:Camera 选项对相机的参数进行获取和设置，支持相机的所有属性节点，如下图红框所示。

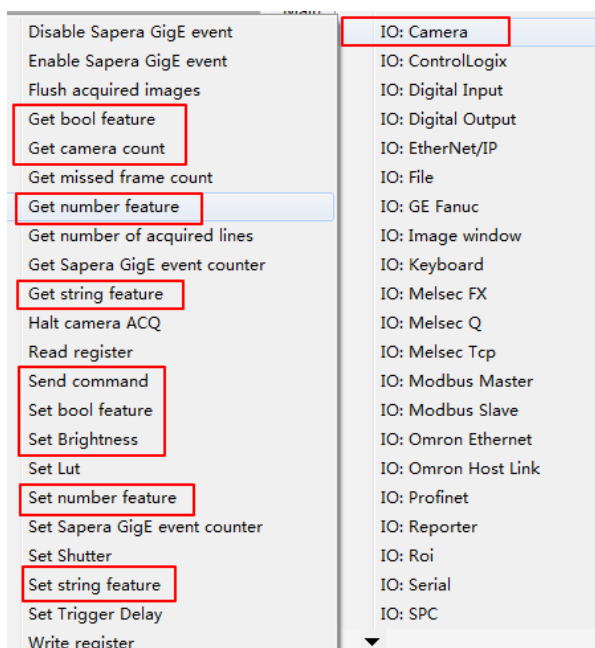


图4-22 IO:Camera 支持的属性节点

