Halcon 二次开发示例程序说明

【摘要】

本文档主要介绍了使用 Halcon 接口进行开发程序方法及过程。在 SDK 开发包目录下，提供了 3 个示例程序，其中两个用 C++开发、一个用 C#开发，三个都为界面程序，分别为 HalconGrabImage 、 Raw2Himage_C 和 Raw2Himage_CSharp 。 Raw2Himage_C 和 Raw2Himage_CSharp 功能完全相同，只是开发的语言不同。这些示例程序分别从不同角度展示了利用 halcondontnet 和 MvCameraControl.Net 进行开发的方法。

本文档就这三个界面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 Halcon 的接口、C++和 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，界面有英文的副本；同时 C#版示例程序也兼容中英文，关键的程序会有中英文的注释，且界面控件也有中英文的区分，可通过切换属性的 language 实现。

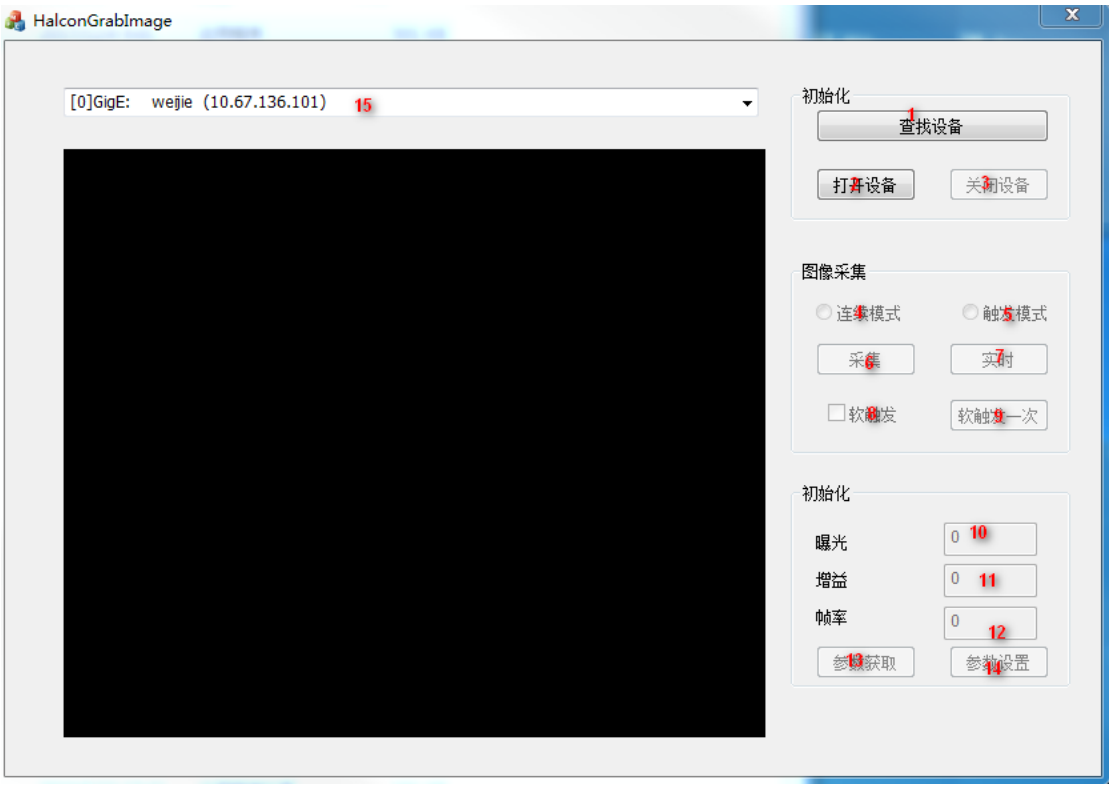
一. HalconGrabImage 使用步骤及开发流程

HalconGrabImage 是一个基本示例程序，包含了 Halcon 常用的一些接口调用，初次使用 Halcon 接口进行二次开发的用户推荐首先参考 HalconGrabImage，其涵盖了大多数用户对 Halcon 接口的使用方法示例需求。

1.1 Demo 软件使用步骤

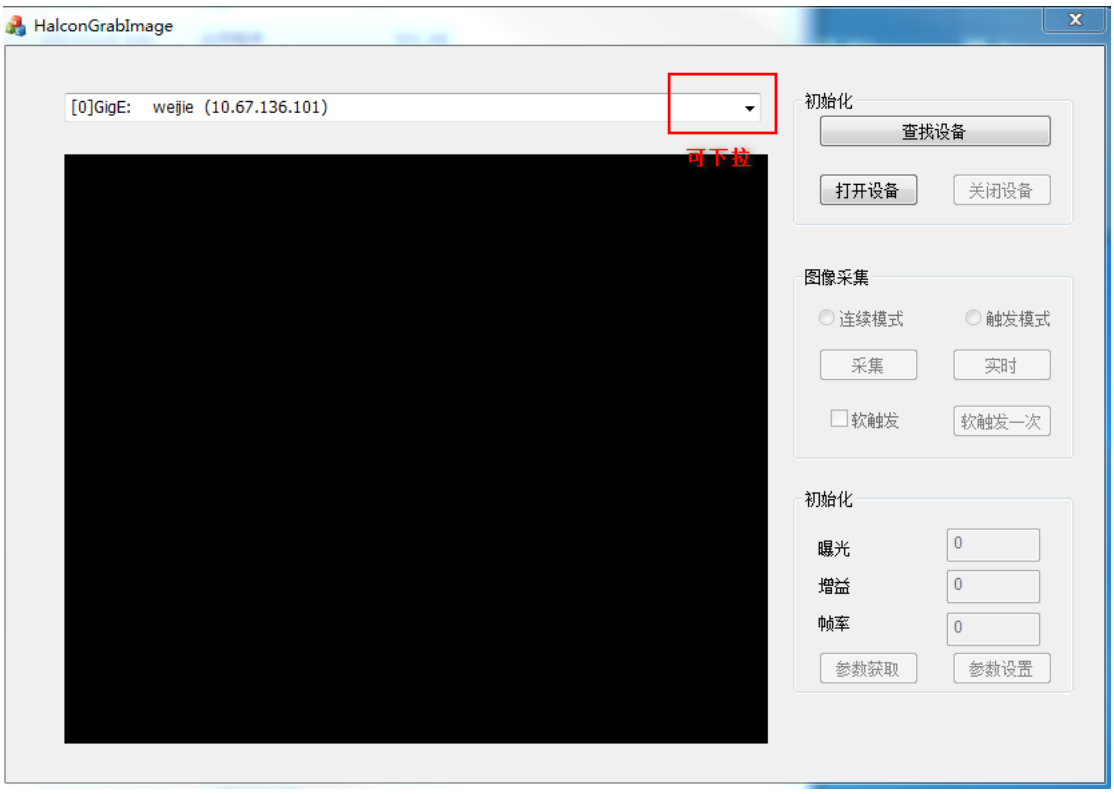
1.1.1 界面总体

软件界面总览，一共包括三个控制模块(初始化，图像采集，参数控制)、一个下拉设备列表和一个图像显示区域

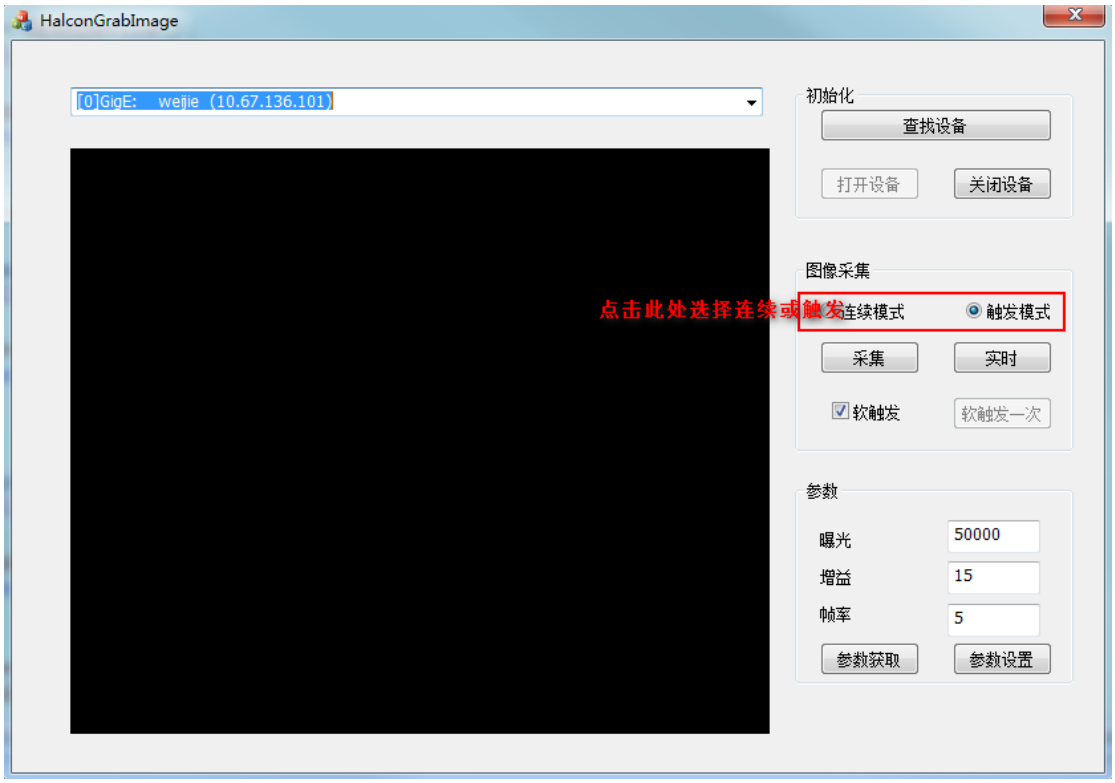


1.1.2 使用过程

点击【查找设备】进行查找设备，这时（15）会出现当前在线的设备列表，命名方式为用户 ID 不为空时显示序列号+设备类型+设备名称+IP 地址，ID 为空时显示为空。



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，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

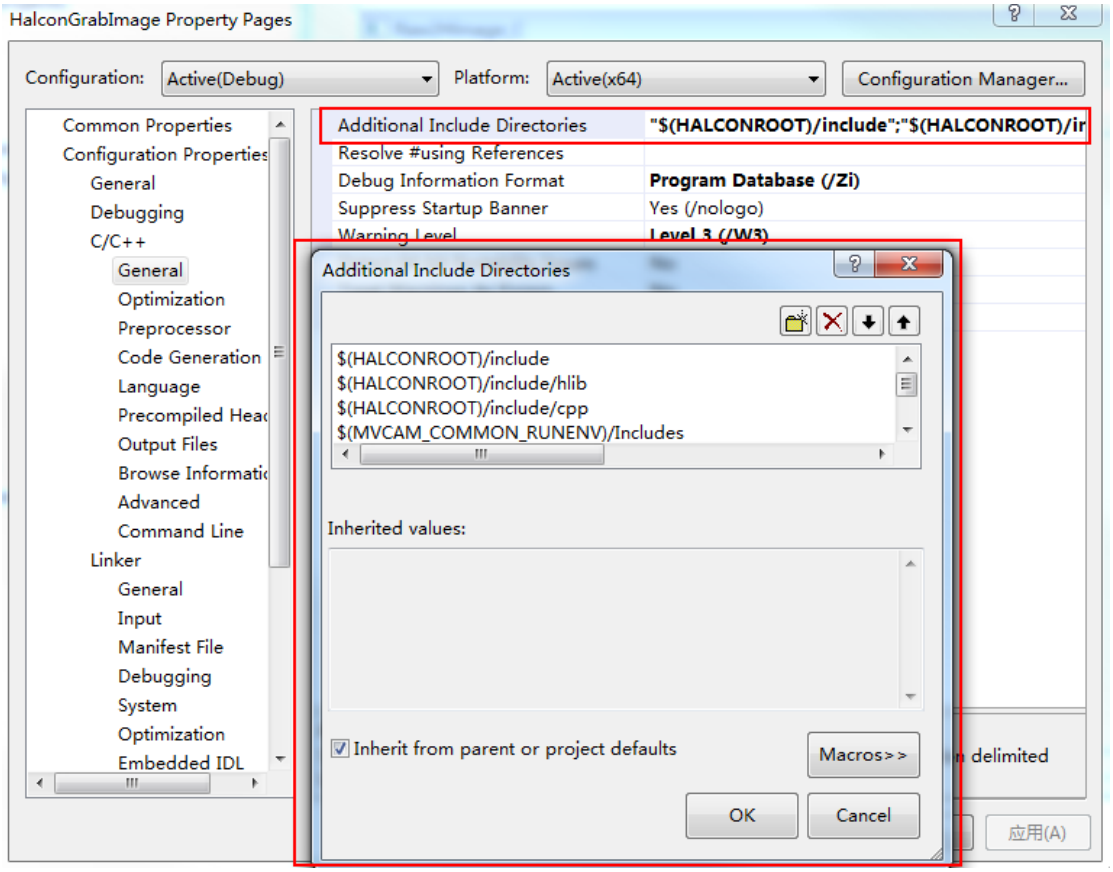
1.2 Demo 软件开发步骤

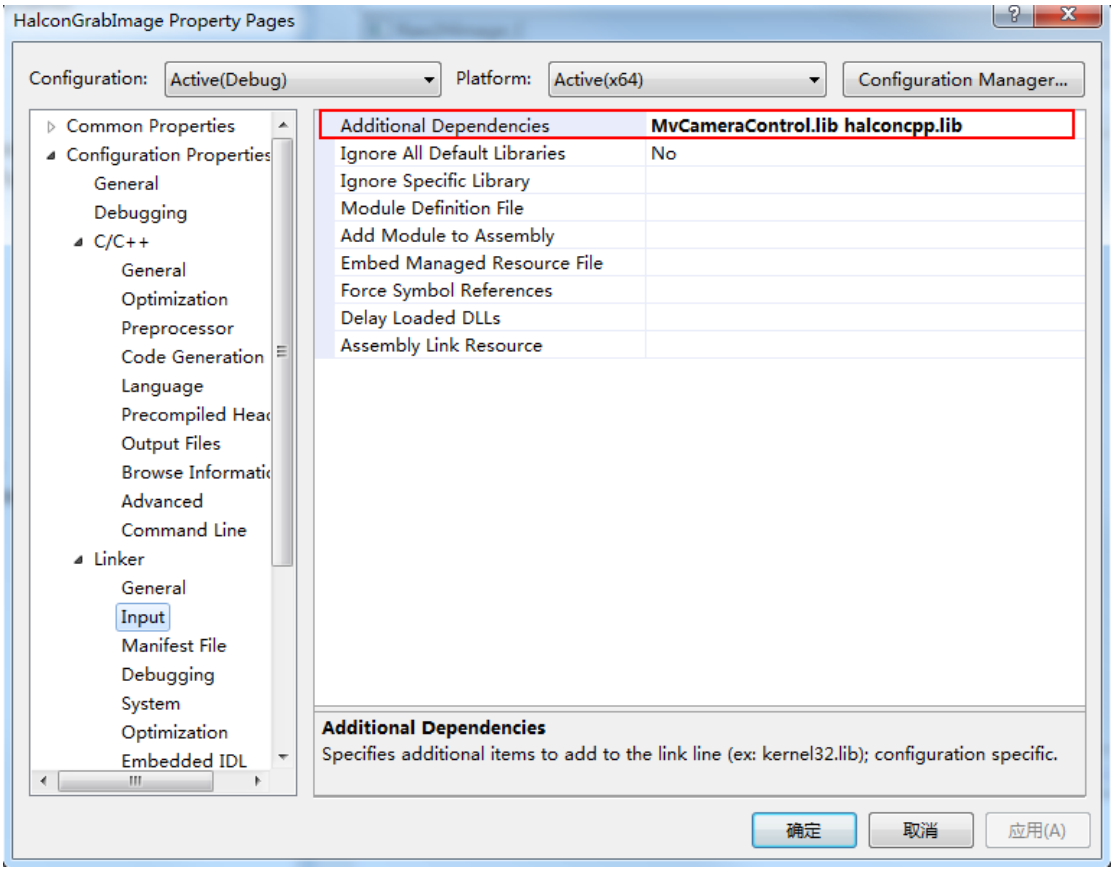
1.2.1 Dll 加载

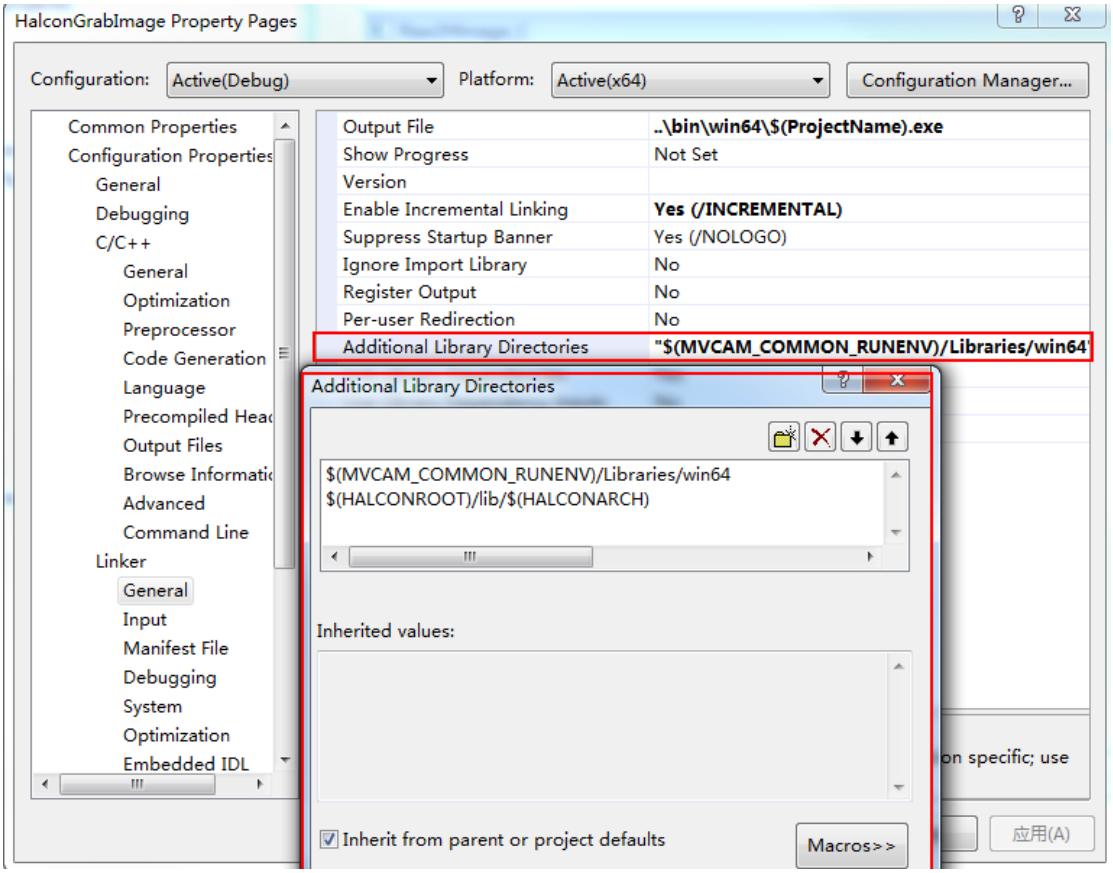
安装好 MVS 和 Halcon 的同时会把相应 32 和 64 的 dll 打到环境变量。

1.2.2 工程配置

创建 C++工程并添加引用，加入 Halcon 和 C++的 SDK 头文件和 lib 文件到工程中，如下图所示。







1.2.3 引用头文件

添加引用后再工程中引用 MvCameraControl.h 和 HalconCpp.h，就可以调 Halcon 和 SDK

中相机操作的函数。

```
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
31 //·实现
32 protected:
33 →   HICON·m_hIcon;
34
35 →   //·生成的消息映射函数
36 →   virtual·BOOL·OnInitDialog();
37 →   afx_msg·void·OnSysCommand(UINT·nID,·LPARAM·lParam);
```

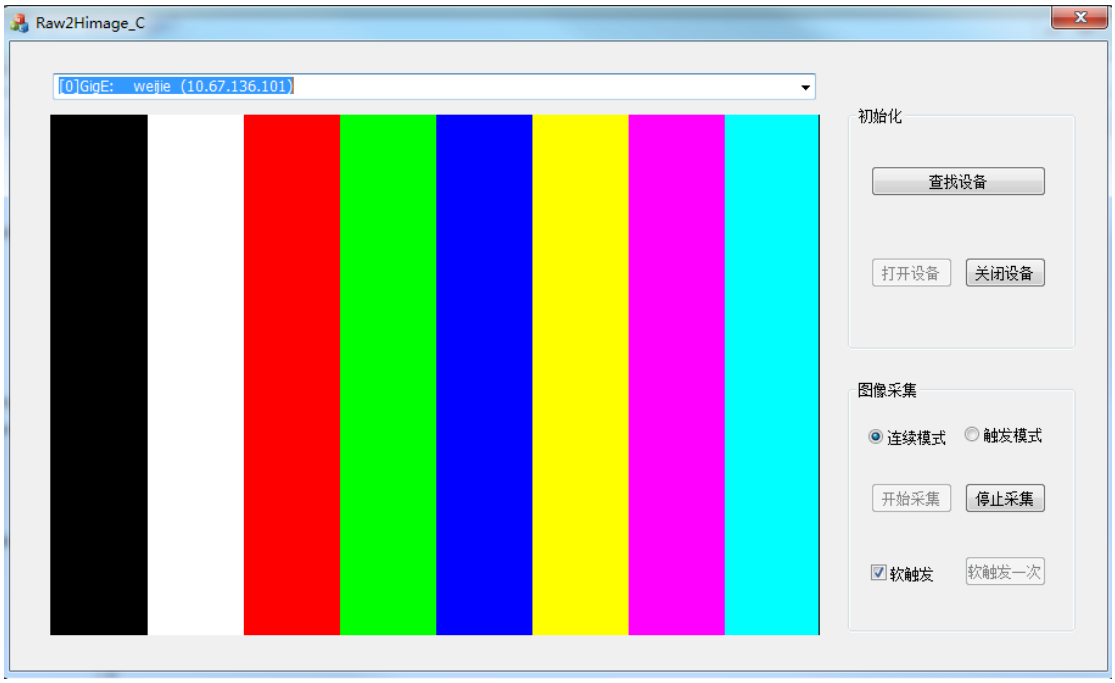
二． Raw2Himage_C 使用步骤及开发流程

Raw2Himage_C 重点展示了使用 Halcon 接口进行格式转换的步骤。告知用户如何使用 Halcon 接口进行图像像素转换以及显示。

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 HalconGrabImage，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

Raw2Himage_C 中，当相机连接时，程序会注册回调函数，图像回调中，当点击取流时相机会自动调用取流函数进行格式转换。

2.2 Demo 软件开发步骤

关于相机操作的开发流程与 HalconGrabImage 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，图像的回调为 RegisterImageCallBack。

首先在 CRaw2Himage_CDlg 类中实现断线重连的函数 ImageCallBack，然后传入回调函数 RegisterImageCallBack 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。当相机开始取流时，程序会进入图像回调中。用户可在图像回调中进行图像格式的转换和显示的操作。注册过程如下：

```
m_pcMyCamera->RegisterImageCallBack(ImageCallBack, this);
```

三．Raw2Himage_CSharp 使用步骤及开发流程

本节介绍的 Demo 重点展示了使用 Halcon 接口进行格式转换的步骤。告知用户如何使用 Halcon 接口进行图像像素转换以及显示，同 Raw2Himage_CSharp 示例程序。

3.1 Demo 软件使用步骤

3.1.1 界面总体

总体界面如下图。界面类似 HalconGrabImage，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。

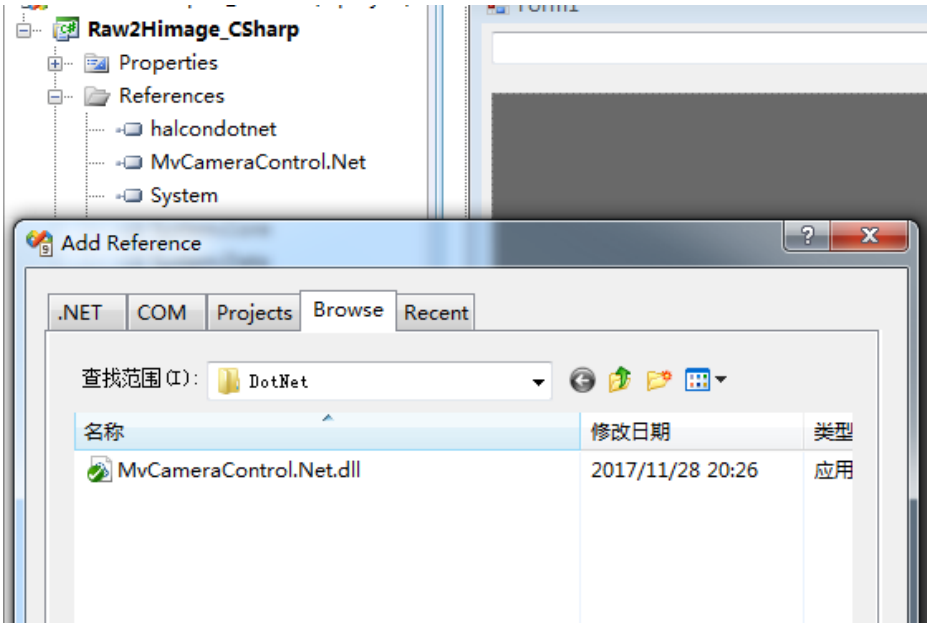
3.2 Demo 软件开发步骤

3.2.1 Dll 加载

安装好 MVS 和 Halcon 的同时会把相应 32 和 64 的 dll 打到环境变量。

3.2.2 工程配置

创建 CS 工程并添加引用，加入 halcondotnet.dll 和 MvCameraControl.Net.dll 到工程中。



3.2.3 引用命名空间

添加引用后再工程中引用命名空间 using MVCameraSDK.NET 和 using HalconDotNet，就可以调 MyCamera 和 Halcon 中相机操作的函数。

```
        public static object ByteToStruct(byte[] bytes, Type type);
        public IntPtr GetCameraHandle();
        public int MV_CC_CloseDevice_NET();
        public int MV_CC_ConvertPixelType_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 nTLayerType, ref MyCamera.MV_CC_DEVICE_INFO_LIST stDevList);
        public static int MV_CC_EnumDevicesEx_NET(uint nTLayerType, 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);
```

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

        public static void AbsDiffImage(HObject image1, HObject image2, out HObject imageAbsDiff, HTuple mult);
        public static void AbsFunct1d(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, HTup
        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);
```

3.2.4 使用相机操作类

Demo 中提供一个封装好的 CameraOperator 类，简化了相机操作的步骤，调用方便并且可扩展性强，用户可以根据需求增加或者修改类的方法。

Demo 中封装了一个 HOperatorSet，通过该类可以直接调用 Haclon 的对应接口，实现像素的转换和显示。

3.2.5 调用过程

在 C#中，用 delegate（代理）的方式代替 C 语言中函数指针。在工业相机 C# SDK 中，图像输出的回调代理为 MyCamera.cbOutputdelegate。

首先在 Form1 类中申明一个回调代理成员变量，如下：

```
MyCamera.cbOutputExdelegate ImageCallback;
```

然后为 ImageCallback 创建一个实例：

```
ImageCallback = new MyCamera.cbOutputExdelegate(GrabImage);
```

其中，GrabImage 表示回调处理函数。

其次，在打开相机操作之后，利用 SDK 中注册回调函数接口，注册回调函数。当程序开始取流后，每当获取到一帧图像数据就会执行回调函数对数据进行自定义的处理。注册过程如下：

```
m_pOperator.RegisterImageCallBack(ImageCallback, IntPtr.Zero);
```

在本示例程序中，GrabImage 函数实现了将获取到的像素数据保存为 Himage 图片格式并显示到控件中。