Machine Vision Camera SDK Demo (Halcon)

User Manual

User Manual

About this Manual

This Manual is applicable to Machine Vision Camera SDK Demo (Halcon).

The Manual includes instructions for using and managing the product. Pictures, charts, images and all other information hereinafter are for description and explanation only. The information contained in the Manual is subject to change, without notice, due to firmware updates or other reasons. Please find the latest version in the company website.

Please use this user manual under the guidance of professionals.

Legal Disclaimer

REGARDING TO THE PRODUCT WITH INTERNET ACCESS, THE USE OF PRODUCT SHALL BE WHOLLY AT YOUR OWN RISKS. OUR COMPANY SHALL NOT TAKE ANY RESPONSIBILITIES FOR ABNORMAL OPERATION, PRIVACY LEAKAGE OR OTHER DAMAGES RESULTING FROM CYBER ATTACK, HACKER ATTACK, VIRUS INSPECTION, OR OTHER INTERNET SECURITY RISKS; HOWEVER, OUR COMPANY WILL PROVIDE TIMELY TECHNICAL SUPPORT IF REQUIRED.

Contents

Overview

This manual mainly introduces the SDK (Software Development Kit) programming methods and procedure of machine vision camera based on Halcon API.
Five Demos are provided in the SDK directory, three of which are developed using C++ and the rest two are in C#, including HalconGrabImage, Raw2Himage_C, Raw2Himage_CSharp, Raw_2_3DFile_C, and Raw_2_3DFile_CSharp. First three demos are interface programs and last two are console programs. Raw2Himage_C and Raw2Himage_CSharp, Raw_2_3DFile_C and Raw_2_3DFile_CSharp have same function, but they are developed based on different language.
The Demos are developed by adopting *halcondontnet* and *MvCameraControl.Net*.
To ensure the proper use of SDK, please refer to the contents below and read the manual carefully before operation and development.

Note

C++ demo is compatible with both English and Chinese and key programs are commented in both languages. The interface has a copy of English. Also, C# demo is the same and language switch can be done between English and Chinese in property of the interface.
Please be noted for C++ programs, a single demo cannot be applied to all Halcon versions due to the difference between them. Thus, applicable Halcon versions are remarked in the project name. For example, HalconGrabImage_10 can be used for Halcon10, HalconGrabImage_11-13 is for Halcon 11-13 and so on.

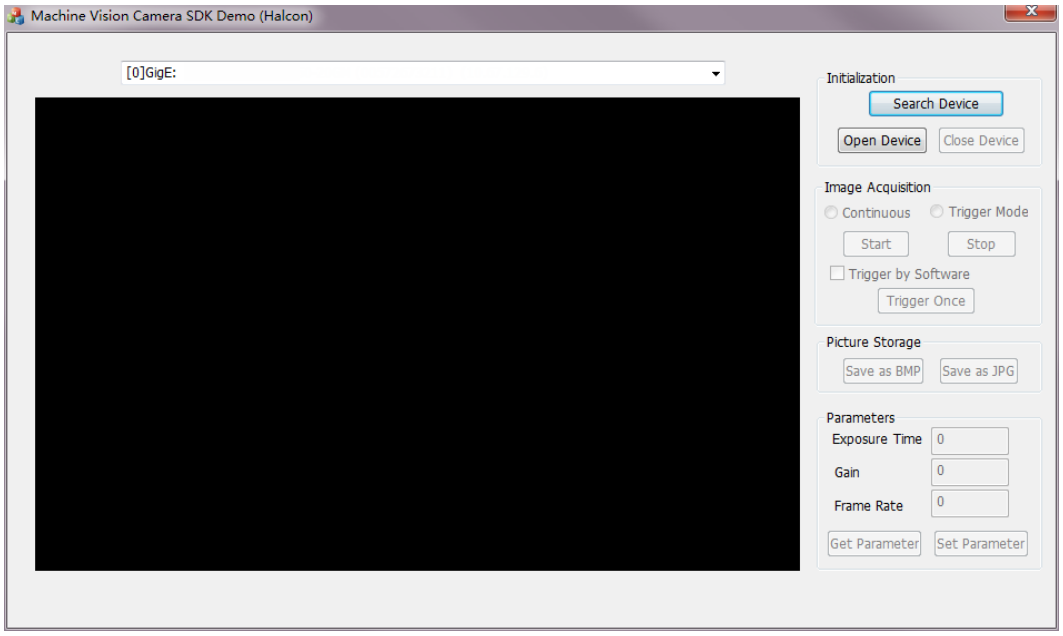
Chapter 1 HalconGrabImage Demo

HalconGrabImage Demo is a basic sample program, which includes general API calling procedure during SDK programming process.
For users who have no experience of SDK programming by Halcon APIs, we recommend the users to refer to the HalconGrabImage Demo, as it contains multiple required examples.

1.1 Demo Introduction

1.1.1 Interface Introduction

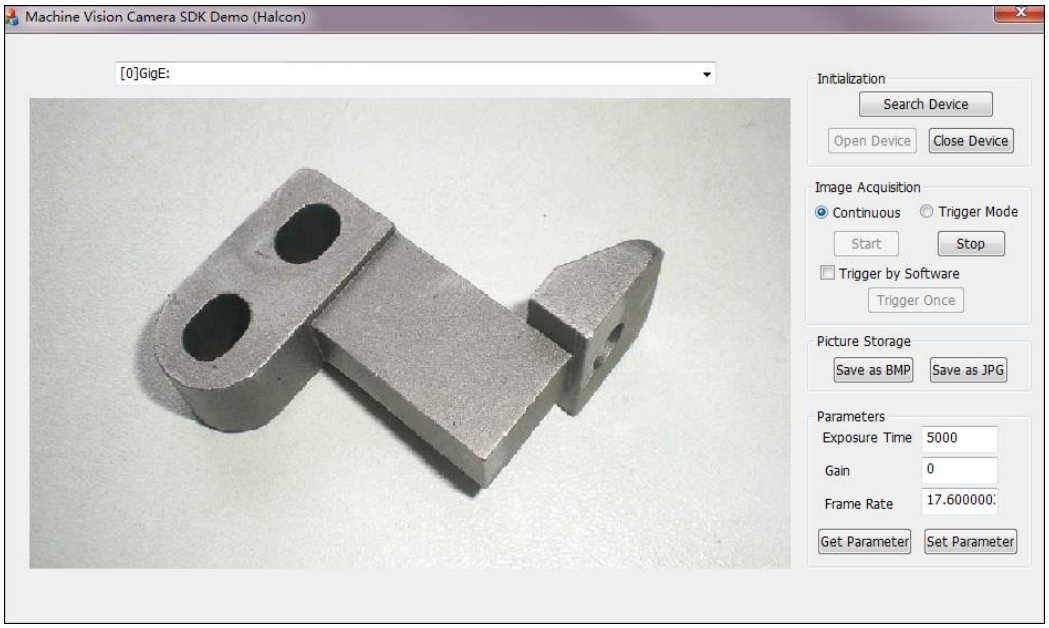
the demo interface includes three control modules (Initialization, Image Acquisition and parameter control), one drop-down list and an image display area.



1.1.2 Operation Procedure

Before running the demo, Halcon plugin (hAcqMVision.dll) should be copied from the Development\ThirdPartyPlatformAdapter under the Client installation path to the Halcon installation location. Please be noted that the plugin used needs to be the corresponding one to the installed Halcon version. If the 64-bit version of Halcon is installed, the plugin should be copied to the 64-bit file. (The same principle is applied to hAcqMVisionxl.dll if Halcon XL is used.)

1. Click **Search Device** in the initialization area to enumerate the online devices and the online devices will be displayed in the drop-down list. Please be noted if the User ID is not blank, the device will be displayed in the format of “Serial N0.” + “Device Type” + “Device Name” + “IP Address”.
2. Select a device in the drop-down list.
3. Click **Open Device** button in the Initialization field to active the Image Acquisition field.
4. Select image acquisition mode as **Continuous** or **Trigger Mode**.
   Please be noted the default image acquisition mode is **Continuous**.
   When **Trigger Mode** is selected, you can check the **Trigger by Software** checkbox. Click **Start** button in the Image Acquisition field to start image acquisition.
   The real-time image will display on the left display window if the **Continuous** mode is selected.
   You can also click **Trigger Once** button to realize software trigger for once if **Trigger by Software** checkbox is checked in Trigger mode.
5. Set the value of exposure time, gain and frame rate in the Parameter field.
6. Click **Set Parameter** button to save the settings.
7. (Optional) You can click **Get Parameter** button in the Parameter field to refresh the value of exposure time, gain and frame rate.
   Please be noted if exception or error occurred during the procedure, the prompt dialog will pop up. Otherwise, the program is running normally.



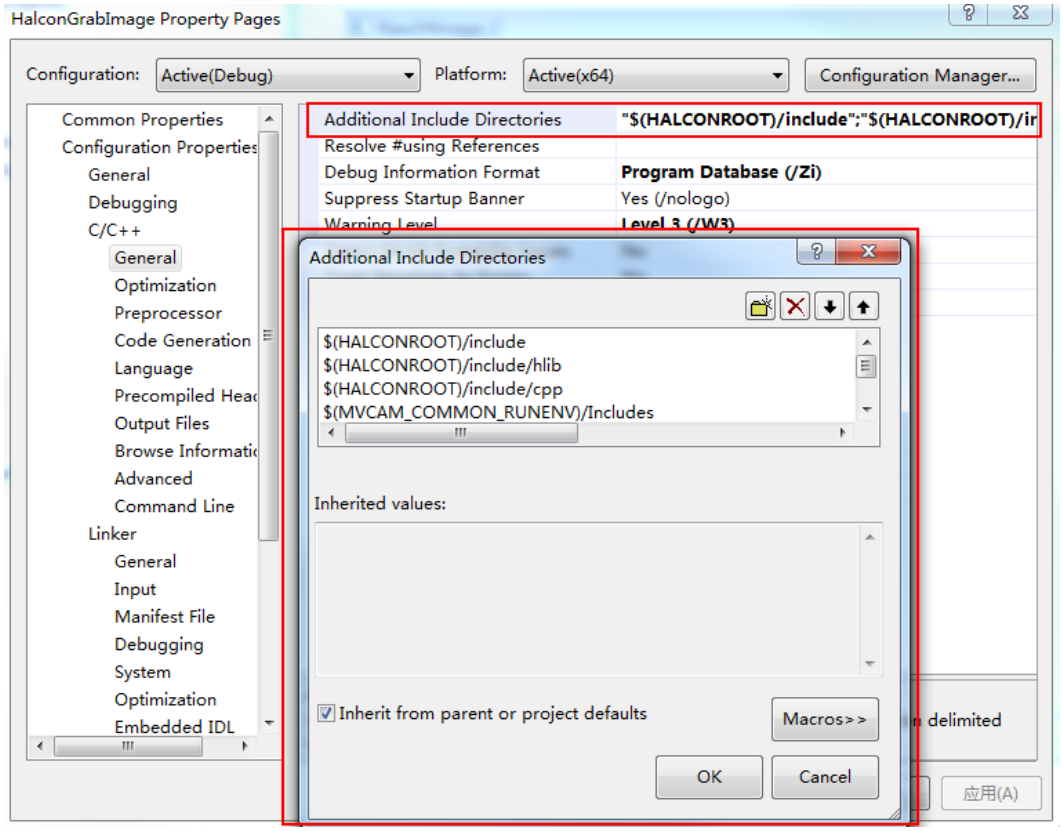
1.2 Development Instruction

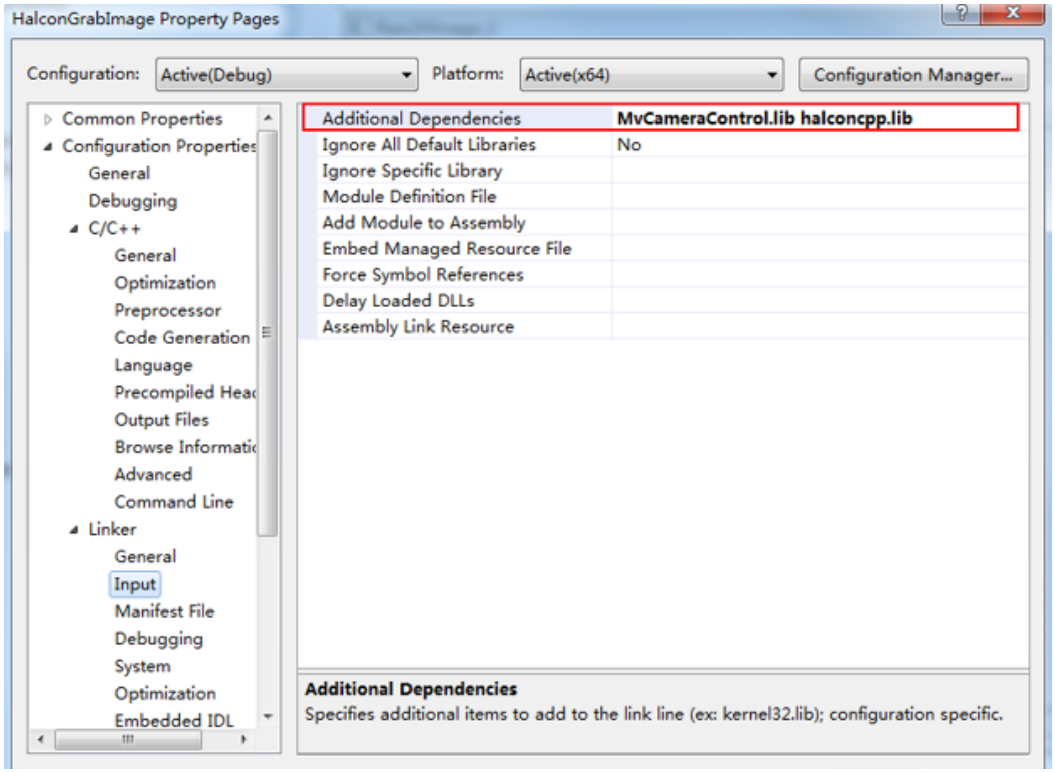
1.2.1 DLL Loading

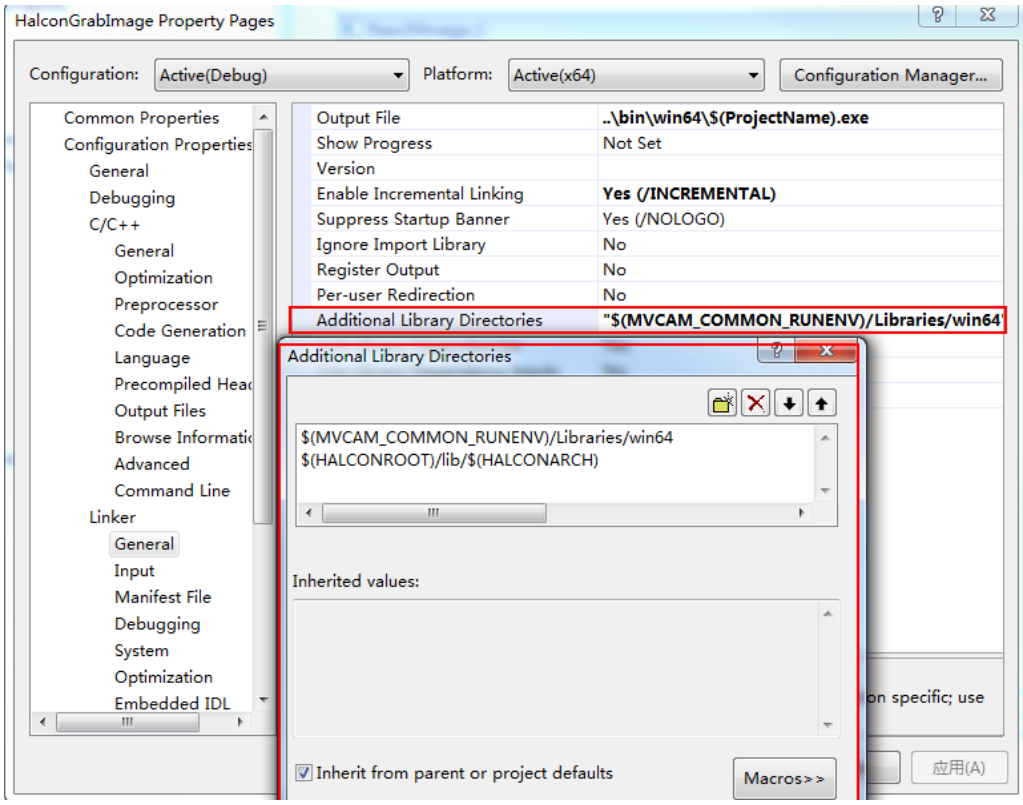
The *.dll* file of 32-bit and 64-bit will be put into the directory of environment variables after installing the Client Application and Halcon.

1.2.2 Project Configuration

Create C++ project and add SDK header file and *.lib* file of Halcon and C++ to the project as reference.


HalconGrabImage Property Pages
Configuration: Active(Debug)
Platform: Active(x64)
Configuration Manager...
Common Properties
Configuration Properties
General
Debugging
C/C++
General
Optimization
Preprocessor
Code Generation
Language
Output Files
Advanced
Command Line
Linker
General
Input
Manifest File
Debugging
System
Optimization
Embedded IDL
Additional Include Directories
"$(HALCONROOT)/include";"$(HALCONROOT)/ir
Resolve #using References
Debug Information Format
Program Database (/Zi)
Suppress Startup Banner
Yes (/nologo)
Additional Include Directories
$(HALCONROOT)/include
$(HALCONROOT)/include/hlib
$(HALCONROOT)/include/cpp
$(MVCAM_COMMON_RUNENV)/Includes
Inherited values:
Inherit from parent or project defaults
Macros>>
OK
Cancel
delimited
应用(A)


HalconGrabImage Property Pages
Configuration: Active(Debug)
Platform: Active(x64)
Configuration Manager...
Common Properties
Configuration Properties
General
Debugging
C/C++
General
Optimization
Preprocessor
Code Generation
Language
Output Files
Advanced
Command Line
Linker
General
Input
Manifest File
Debugging
System
Optimization
Embedded IDL
Additional Dependencies
MvCameraControl.lib halconcpp.lib
Ignore All Default Libraries
No
Ignore Specific Library
Module Definition File
Add Module to Assembly
Embed Managed Resource File
Force Symbol References
Delay Loaded DLLs
Assembly Link Resource
Additional Dependencies
Specifies additional items to add to the link line (ex: kernel32.lib); configuration specific.



1.2.3 Header Reference

Referring the naming space *MvCameraControl.h* and *HalconCpp.h* in the project to call the camera operation function of Halcon and SDK

```
//·HalconGrabImageDlg.h·:·头文件
//

#pragma·once
#include·"afxwin.h"
#include·"MvCameraControl.h"
#include·"HalconCpp.h"·

using·namespace·Halcon;

/*函数返回码定义*/
typedef·int·Status;
#define·STATUS_OK··············0
#define·STATUS_ERROR···········-1

//·CHalconGrabImageDlg·对话框
class·CHalconGrabImageDlg·:·public·CDialog
{
//·构造
public:
→   CHalconGrabImageDlg(CWnd*·pParent·=·NULL);→ //·标准构造函数

//·对话框数据
→   enum·{·IDD·=·IDD_HALCONGRABIMAGE_DIALOG·};

→   protected:
→   virtual·void·DoDataExchange(CDataExchange*·pDX);→   //·DDX/DDV·支持


//·实现
protected:
→   HICON·m_hIcon;

→   //·生成的消息映射函数
→   virtual·BOOL·OnInitDialog();
→   afx_msg·void·OnSysCommand(UINT·nID,·LPARAM·lParam);
```

Chapter 2 Raw2Himage_C Demo

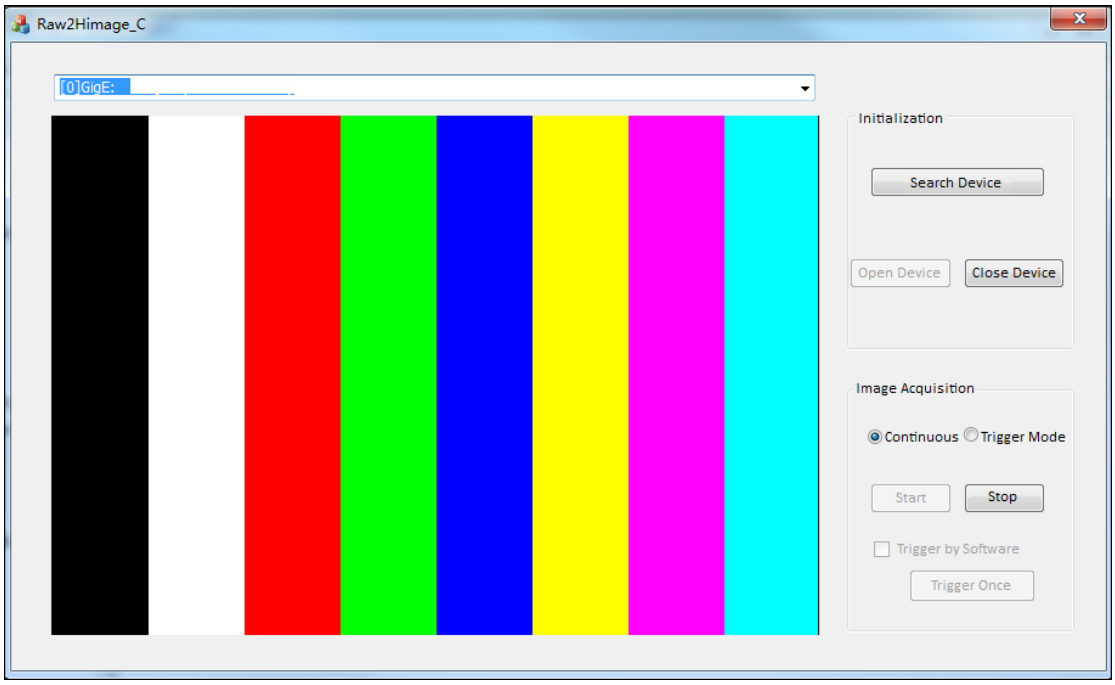
The Raw2Himage_C Demo mainly introduces the operations of format transformation via Halcon APIs.

The Demo describes the process of image pixel transformation and image display.

2.1 Demo Introduction

2.1.1 Interface Introduction

The interface of Raw2Himage_C Demo is similar with that of HalconGrabImage Demo. The Raw2Himage_C Demo can realize the following functions: Search Device, Open/Close device, Start/Stop acquisition and Set Trigger.



2.1.2 Operation Procedure

For Raw2Himage_C Demo, when connecting to the camera, image acquisition function is called to create a thread for image data acquisition and image format transformation after starting acquisition.

2.2 Development Instruction

The development procedure of this demo is similar to that of HalconGrabImage Demo, please refer to *Chapter 1.2 Development Instruction* for details. The following steps mainly introduces the application method of creating thread to acquire image data.

Steps:

1. Call API MV_CC_StartGrabbing of SDK to start for image acquisition.
2. Create a WorkThread.
3. Call API MV_CC_GetOneFrameTimeout of SDK repeatedly in the created thread.
4. Convert the image data and save in Himage format.
5. Call the Halcon API HalconDisplay to display the image.

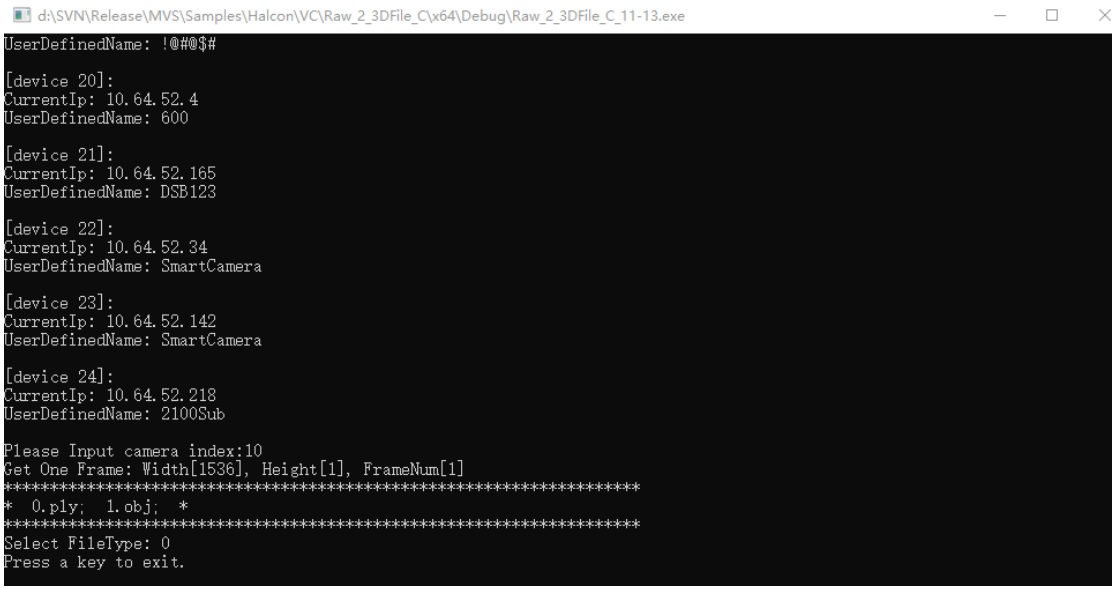
Chapter 3 Raw_2_3DFile_C&Raw_2_3D File_CSharp Demo

Raw_2_3DFile_C and Raw_2_3DFile_CSharp mainly introduces the steps to convert image format using Halcon API and instructs the user to convert 3D image data.

3.1 Demo Instruction

3.1.1 Interface Introduction

The console program includes the following functions: Search Device, Open/Close Device, Start/Stop Acquisition and Save Halcon3D Image



3.1.2 Operation Procedure

In Raw_2_3DFile_C and Raw_2_3DFile_CSharp demos, after the cameras are enumerated, the input of subscript of 3D devices would trigger the action of image acquisition function in SDK for acquiring and converting image data.

3.2 Development Instruction

The development procedure is similar as that of the HalconGrabImage demo. This part mainly introduces the steps of converting 3D data from camera to 3D data format in Halcon and save in the format of *.ply* or *.obj.*

First call StartGrabbing interface from SDK and get data of one frame via MV_CC_GetOneFrameTimeout. After getting the image data, format transform can be done and be saved as *.ply* or *.obj* for Halcon.

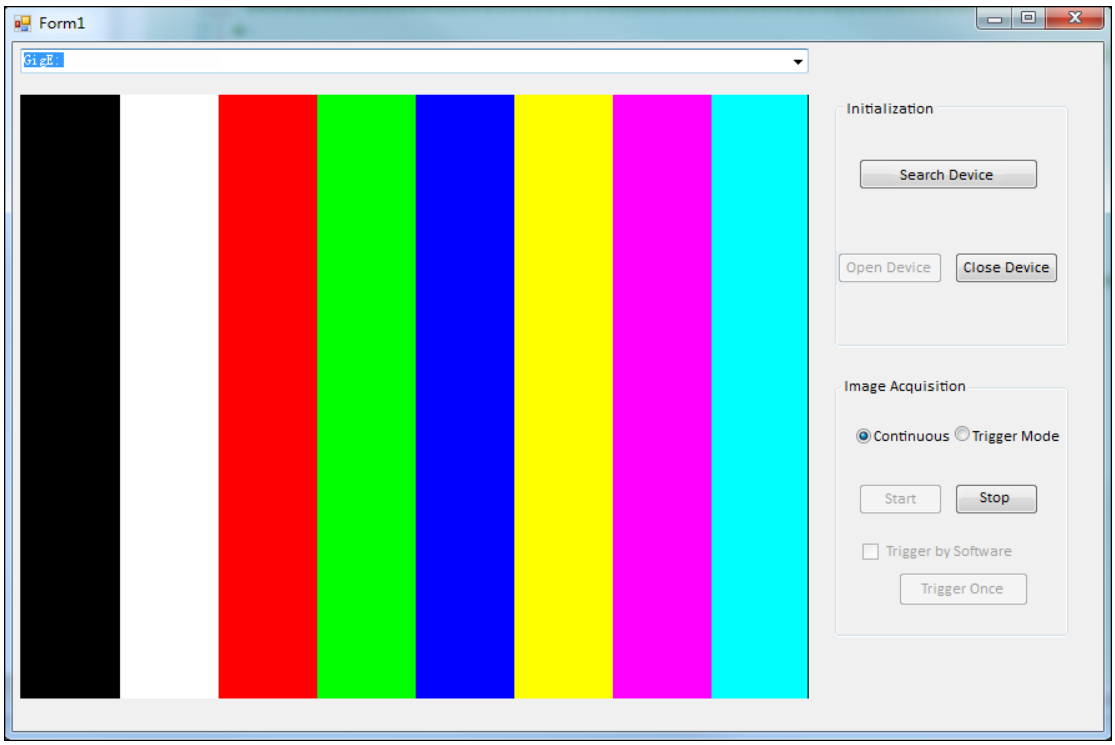
Chapter 4 Raw2Himage_CSharp Demo

The Demo in this section mainly realizes the format transformation via Halcon APIs.

4.1 Demo Instruction

4.1.1 Interface Introduction

This demo is similar to HalconGrabImage and also includes the following functions: Search Device, Open/Close Device, Start/Stop Device and Set Trigger.



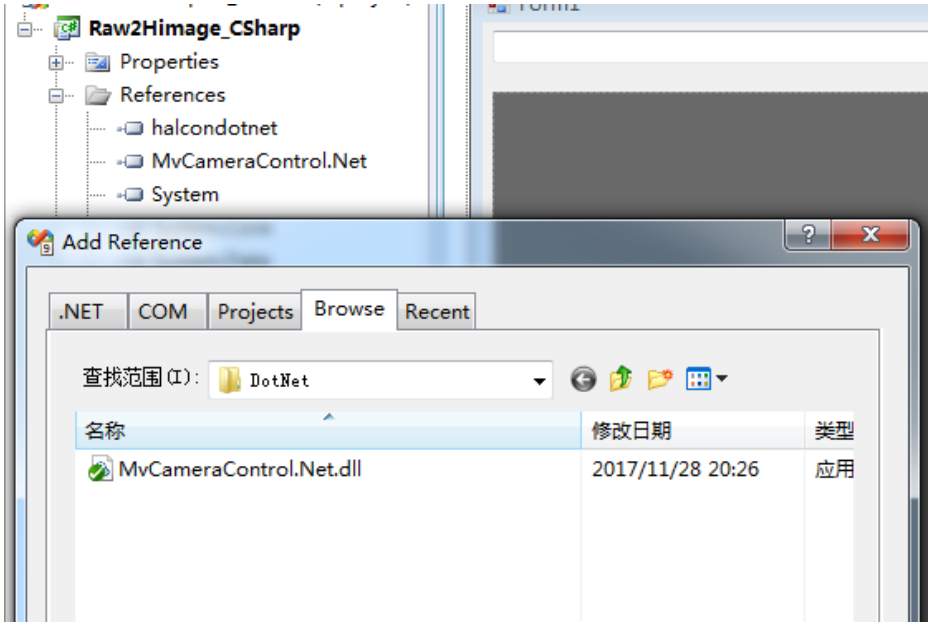
4.2 Development Procedure

4.2.1 DLL loading

The *.dll* file of 32-bit and 64-bit will be put into the directory of environment variables after installing the Client Application and Halcon.

4.2.2 Project Configuration

Create CS Project and add halcondotnet.dll and MvCameraControl.Net.dll to the project.



4.2.3 Namespace Reference

Referring the naming space *using MVCameraSDK.NET* and *using HalconDotNet* in the project to call the camera operation function of *My Camera* and *Halcon*.

```
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);
```

4.2.4 Operation Procedure

Steps:

1. Call API MV_CC_StartGrabbing_NET of SDK to start acquisition.
2. Create a ReceiveImageWorkThread.
3. Call API MV_CC_GetOneFrameTimeout_NET of SDK repeatedly in the created thread to get image.
4. Convert the image format and save it as Himage format.
5. Call the Halcon API HalconDisplay to display the image.