# 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

# Chapter 1 Overview

This manual mainly introduces the SDK (Software Development Kit) programming methods and procedure of machine vision camera based on Halcon API.

Three Demos are provided in the SDK directory, including HalconGrabImage, Raw2Himage_C, and Raw2Himage_CSharp. All the demos are interface programs, first two are developed based on C++ language, and last one based on C# language. Raw2Himage_C and Raw2Himage_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.

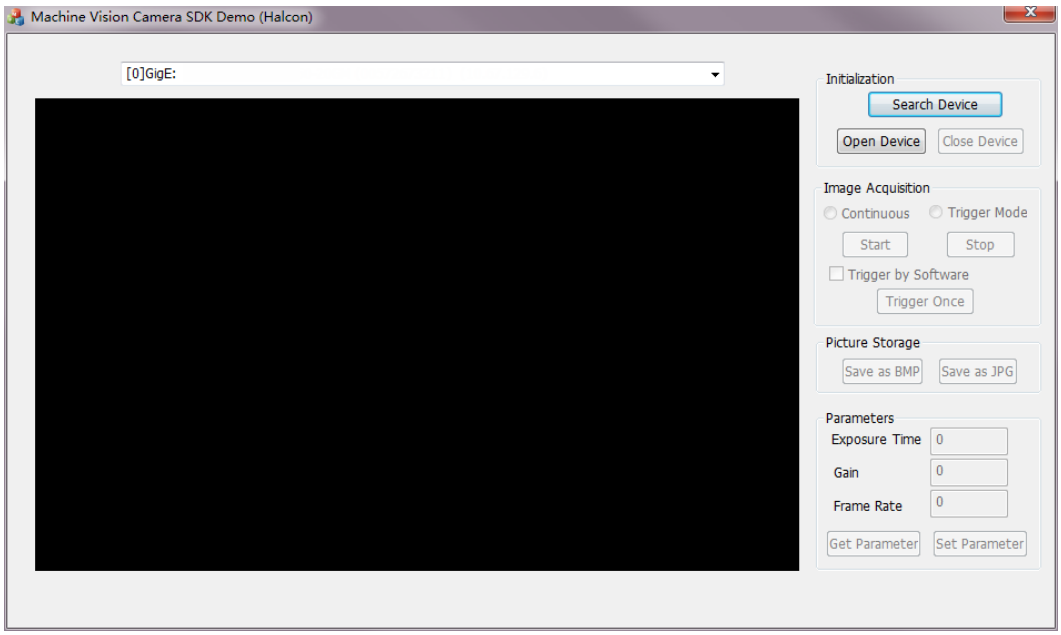
# Chapter 2 HalconGrabImage Demo

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.

## 2.1 Interface Overview

The interface of HalconGrabImage Demo is as following.

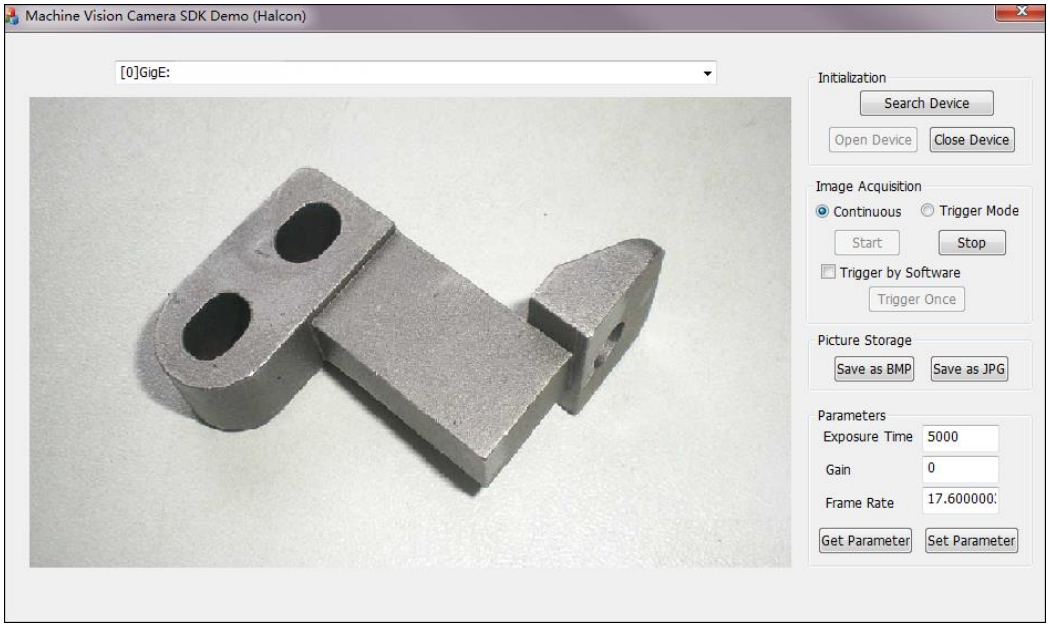



## 2.2 Operation Procedure

***Steps:***

1. Click **Search Device** in the Initialization field to search the online device.
   The online devices will display in the drop-down list of the upper left corner field.
   ***Note:*** If the user ID is not empty, the devices will be displayed as “serial No.” + “device type” + “device name” + “IP address”; otherwise nothing will be displayed..
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**.
   ***Notes:***
   - The default image acquisition mode is **Continuous**.
   - When **Trigger Mode** is selected, you can check the **Trigger by Software** checkbox.
5. 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.




6. Set the value of exposure time, gain and frame rate in the Parameter field.
7. Click **Set Parameter** button to save the settings.
8. (Optional) You can click **Get Parameter** button in the Parameter field to refresh the value of exposure time, gain and frame rate.

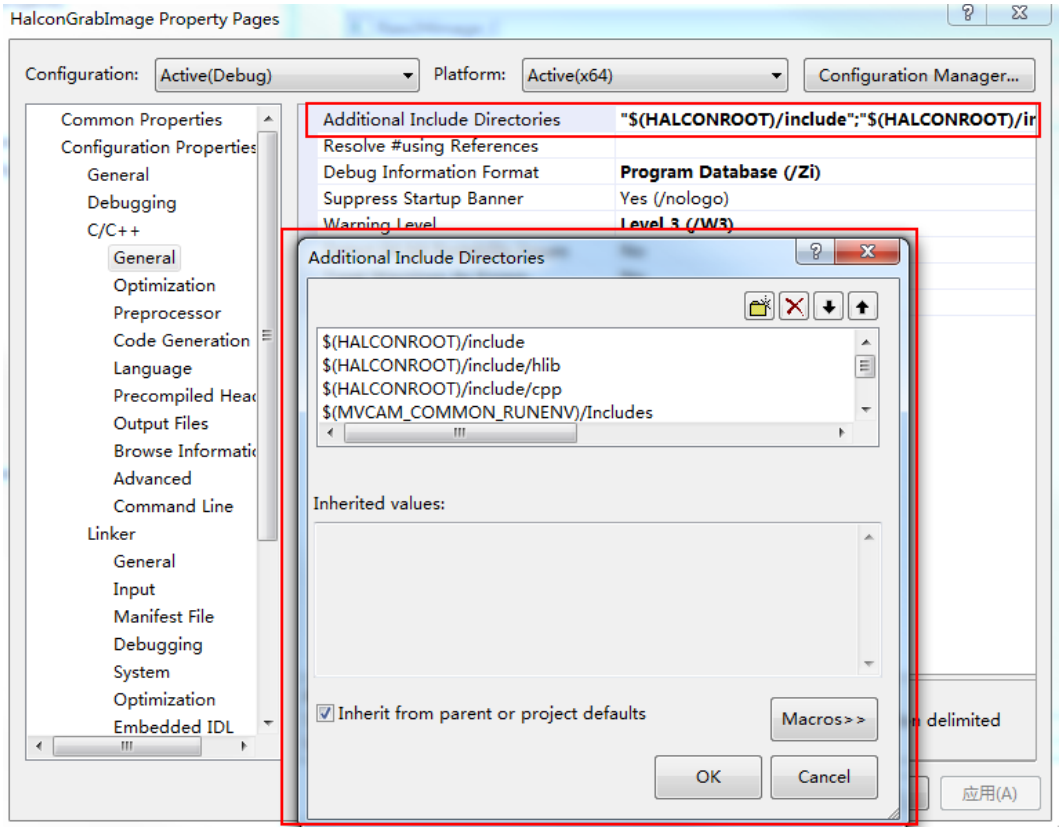
   ***Note:*** If exception or error occurred during the procedure, the prompt dialog will pop up.

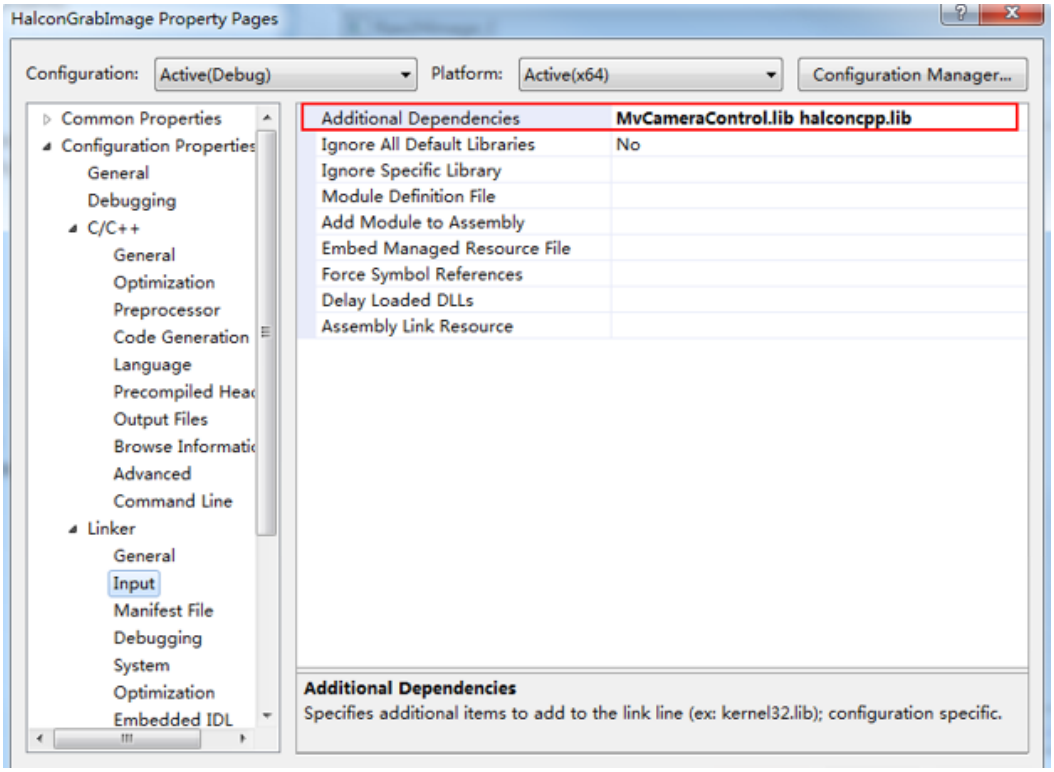
## 2.3 Programming Guideline

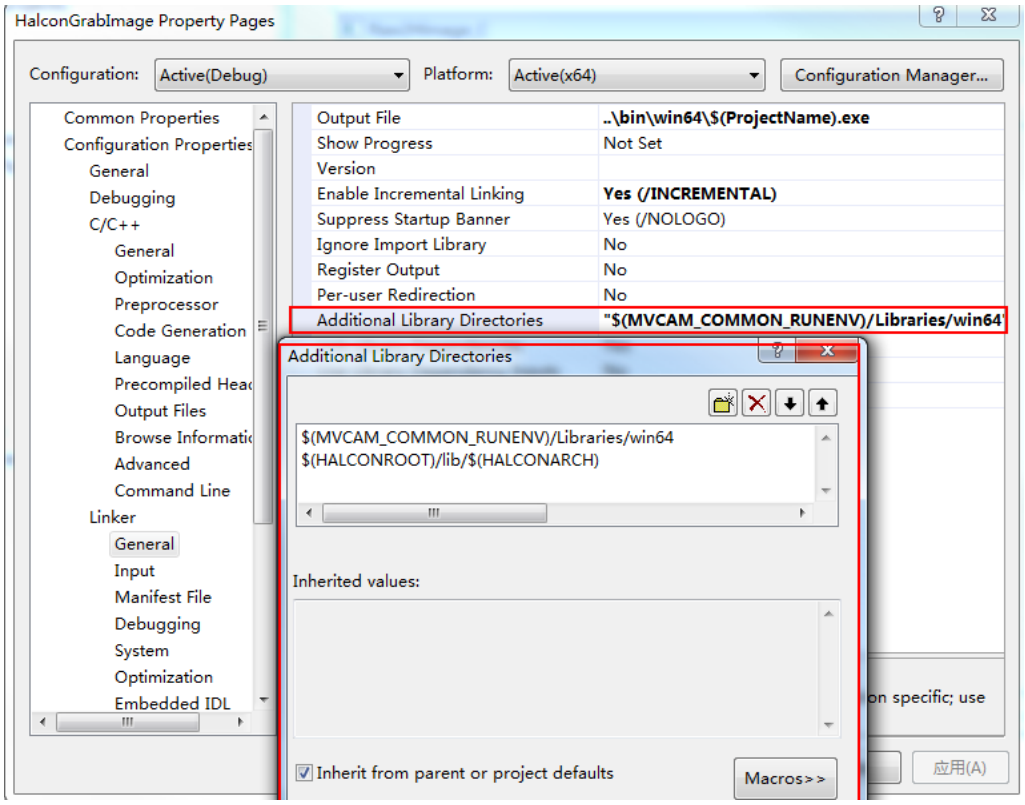
***Steps:***

1. Load DLL.

   The *.dll* file of 32-bit and 64-bit will be put into the directory of environment variables after installing the MVS and Halcon.
2. Configure project.
   1) Create C++ project.
   2) Add SDK header file and *.lib* file of Halcon and C++ to the project.


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
Precompiled Hea
Output Files
Browse Informati
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)
Warning Level
Level 3 (/W3)
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
Precompiled Hea
Output Files
Browse Informati
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.



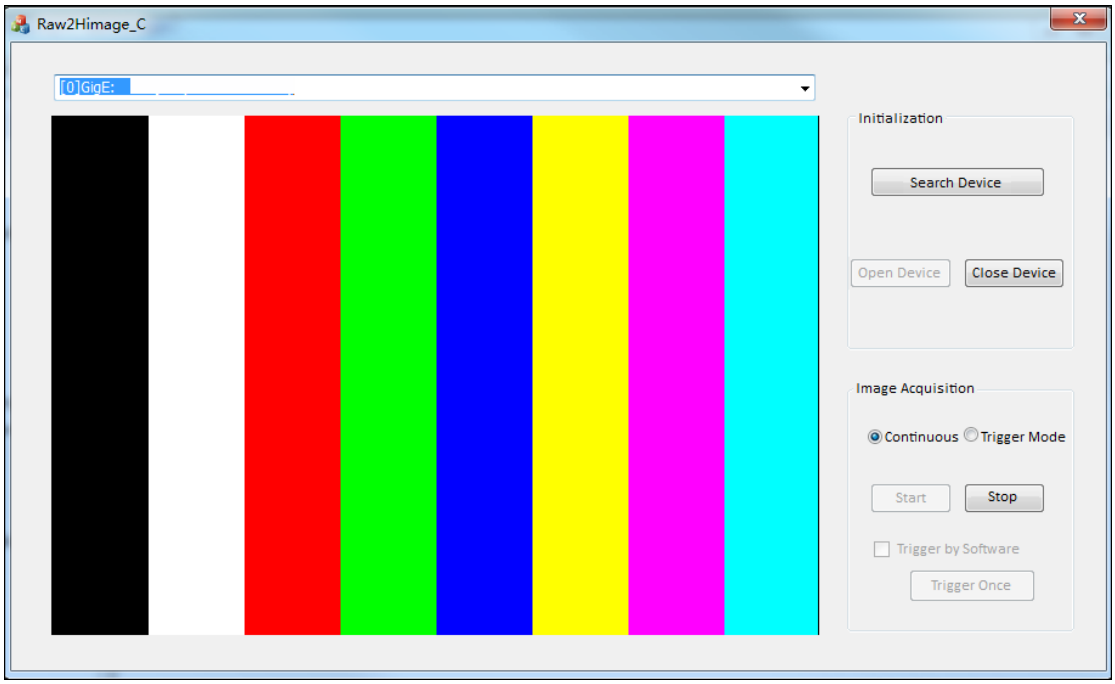

3. Reference the naming space *MvCameraControl.h* and *HalconCpp.h* in the project to call the camera operation function of Halcon and SDK

# Chapter 3 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.

## 3.1 Interface Overview

The interface of Raw2Himage_C Demo is similar with that of HalconGrabImage Demo. The Raw2Himage_C Demo can realize the functions: search device, control device, start acquisition, stop acquisition, and set trigger.




## 3.2 Operation Procedure

The operation procedure of Reconnect Demo is similar with that of HalconGrabImage Demo, please refer to *Chapter 2.2 Operation Procedure* for details.

***Note:***

For Raw2Himage_C Demo, when connecting the camera, it will register the callback function, and the camera will automatically call streaming function to transform format after starting streaming.

## 3.3 Programming Guideline

The programming guidance of Reconnection Demo is similar with that of HalconGrabImage Demo, please refer to *Chapter 2.3 Programming Guidance* for details. Here we introduce the application method of callback function.

For C++ language, you should realize the callback function via transmitting function pointer. So the image callback is RegisterImageCallBack in the machine vision camera SDK (C++).

***Steps:***

1. Realize the reconnection function ImageCallBack in CRaw2Himage_CDlg class.
2. Transmit to callback function RegisterImageCallBack.
3. Register callback function by calling the callback function registration API after opening the camera.

   ***Example:*** m_pcMyCamera->RegisterImageCallBack(ImageCallBack, this);

***Note:***

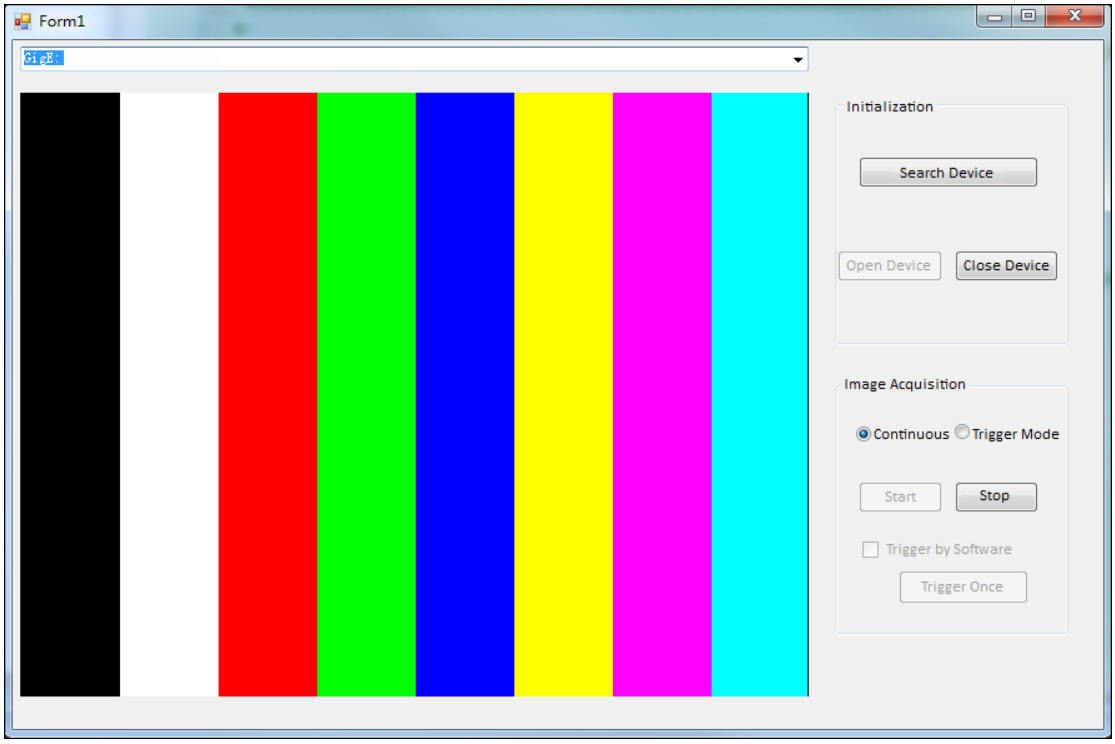
When the camera starts streaming, the demo starts image callback, and you can realize the operations of transforming image format and displaying in image callback.

# Chapter 4 Raw2Himage_CSharp Demo

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

## 4.1 Interface Overview

The Raw2Himage_CSharp Demo for machine vision camera can realize the function of device search, device control, image acquisition and set trigger.




## 4.2 Operation Procedure

***Steps:***

1. Load DLL.

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

2. Configure project.

   3) Create CS project.

   4) Add *halcondotnet.dll* and *MvCameraControl.Net.dll* to the project.

3. Reference 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);
```

# 4.3 Camera Operation Class

An encapsulated *CameraOperator* class is provided in the Demo, which simplifies the camera operations. It has good extendibility and can be called easily.

An encapsulated *HOperatorSet* class is provided in the Demo, you can call corresponding Halcon APIs via this class to realize pixel transforming and displaying.

# 4.4 Calling Procedure

For C# language, you should replace the function pointer of C language by delegate (proxy) method. So the image output callback proxy is *MyCamera.cbOutputdelegate* in the machine vision camera SDK

(C#).

***Steps:***

1. Assign a variable for callback proxy member in Forml class.
   ***Example:*** MyCamera.cbOutputExdelegate ImageCallback;
2. Create an example for ImageCallback.
   ***Example:*** ImageCallback = new MyCamera.cbOutputExdelegate(GrabImage);
   While, the GrabImage indicates the callback handling function.
3. Register callback function by calling the callback function registration API after opening the camera.
   ***Example:*** m_pOperator.RegisterImageCallBack(ImageCallback, IntPtr.Zero);

When the Demo starts streaming, it will handle the data of each received frame image data by executing callback function.

In the Demo, function *GrabImage* realized the function of saving received pixel data as picture in Himage format, and displaying it.