Machine Vision Camera SDK Demo (C#)

User Manual

User Manual

About this Manual

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

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

Contents

Chapter 1 Overview

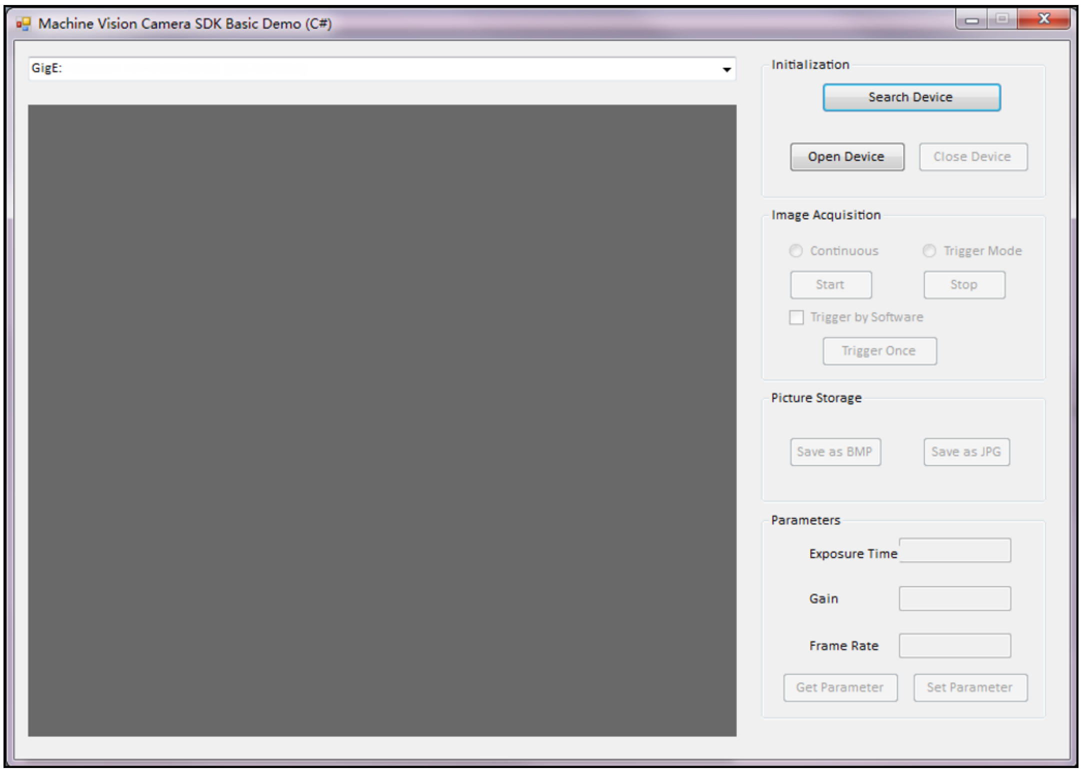
This manual mainly introduces the SDK (Software Development Kit) programming methods and procedure of machine vision camera based on C# language.
Twenty-one C# Demos are provided in the SDK directory, including six Form Demos and eighteen Console Demos.
Form Demos: BasicDemo, ReconnectDemo, SetIODemo, ForceIPDemo, MultipleDemo, and BasedOnGenTL.
Console Demos: CamLBasicDemo, ChunkData, ConnectSpecCamera, ConvertPixelType, Events, Grab_ActionCommand, Grab_Callback, GrabImage, GrabStrategies, MultiCast, Recording, SpatialDenoise ParametrizeCamera_FileAccess, ParametrizeCamera_LoadAndSave SavePonitCloudData_3D , ImageEnhance, LensShadingCorrection and ColorCorrect.
All the Demos are developed by adopting 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 BasicDemo

BasicDemo is a basic sample program, which includes general API calling procedure during SDK programming process.
For users who have no SDK programming experience of machine vision camera, we recommend the users to refer to the BasicDemo, as it contains multiple required examples.

2.1 Interface Overview

The BasicDemo based on C# language for machine vision camera can realize the function of image display, initialization, image acquisition, picture storage and parameter control.



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 “device type” + “device name” + “serial No.” + “IP address”; otherwise the devices will be displayed as “device type” + “device model” + “serial No.” +”IP address”.
2. Click to 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.**

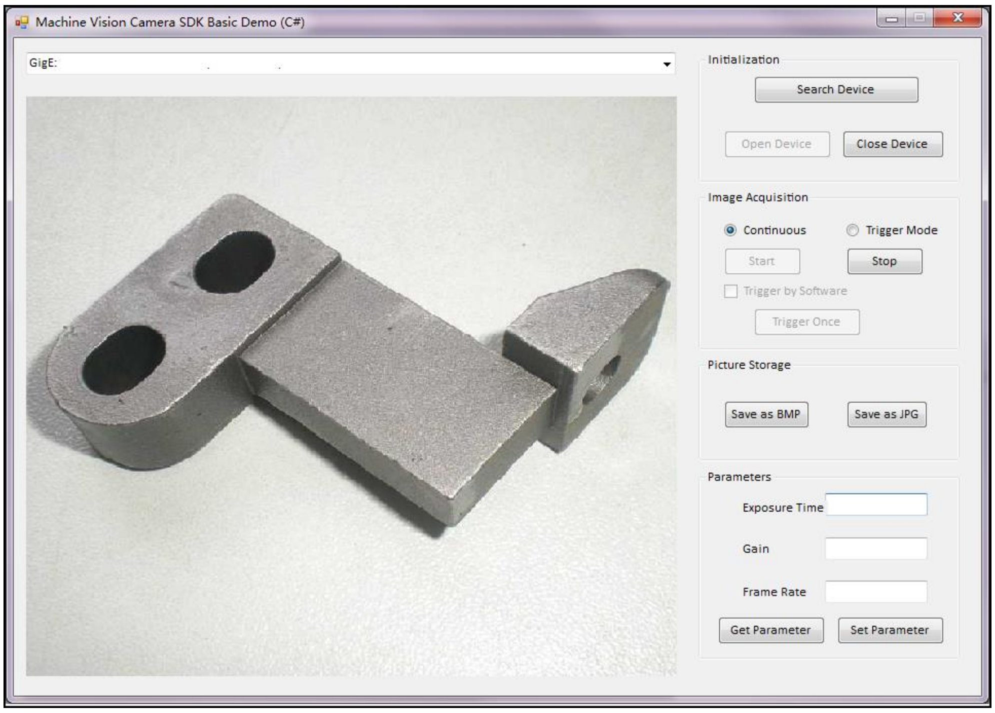
Notes:

- 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. Click **Save as BMP** or **Save as JPG** button in the Picture Storage field to save the current image, which is named by **.bmp* or **.jpg*, to the directory of .exe.
7. Set the value of exposure time, gain and frame rate in the Parameter field.
8. Click **Set Parameter** button to save the settings.
9. (Optional) You can click **Get Parameter** button in the Parameter field to refresh the value of exposure time, gain and frame rate.

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

2.3 Programming Guideline

Steps:

1. Load DLL.

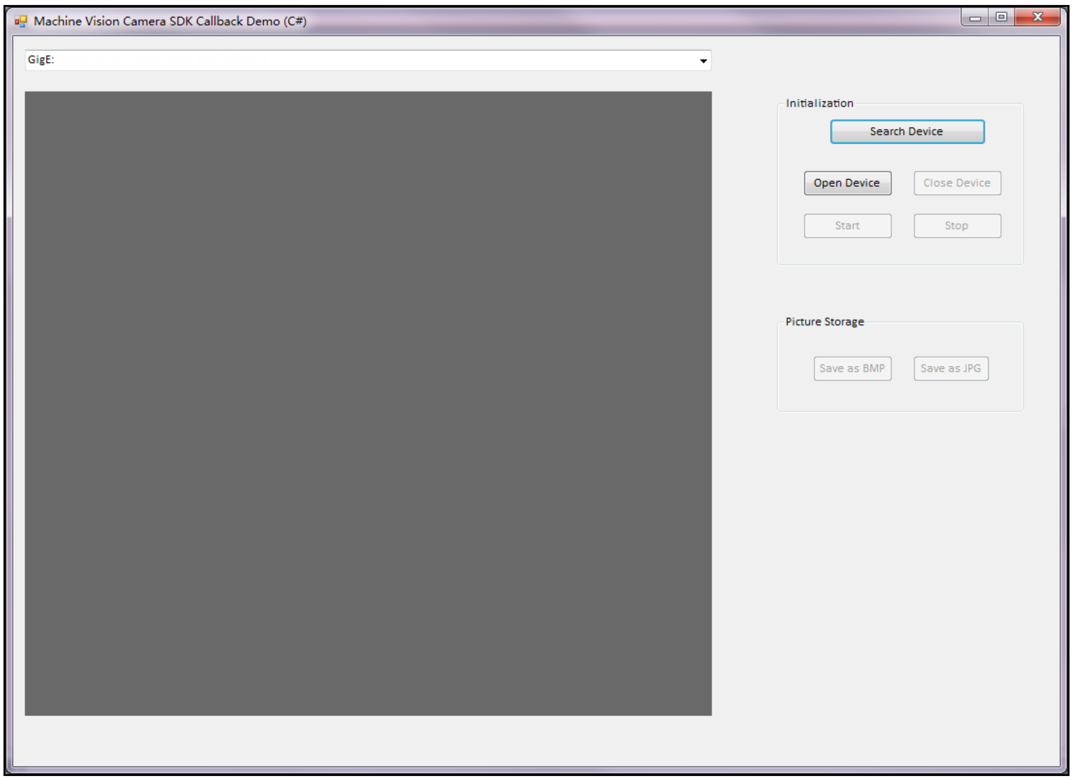
   The *.dll* file of 32-bit and 64-bit will be put into the directory of environment variables after installing the MVS.
2. Configure project.
   1) Create CS project.
   2) Add *MvCameraControl.Net.dll* to the project.
3. Reference the naming space *using MvCamCtrl.NET* in the project to call the camera operation function of *My Camera*.

Chapter 3 ReconnectDemo

The ReconnectDemo mainly introduces the operations of disconnected camera reconnection in the SDK.
The following sample program describes the process of disconnection callback and camera reconnection method.

3.1 Interface Overview

The ReconnectDemo based on C# language for machine vision camera can realize the function of device search, device control, image acquisition and configuration trigger.



3.2 Operation Procedure

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

3.3 Programming Guideline

The programming guidance of Reconnection Demo is similar with that of BasicDemo, please refer to *Chapter 2.3 Programming Guidance* for details. Here we introduce the application method of callback function.
For C# language, you should replace the function pointer of C language by delegate (proxy) method. So the exceptional disconnection callback proxy is *MyCamera. cbExceptiondelegate* in the machine vision camera

SDK (C#).

Steps:

1. **Assign a variable for callback proxy member in FormI class.**

   Example: **MyCamera.cbOutputdelegate pCallBackFunc;**
2. **Create an example for pCallBackFunc.**

   Example: **pCallBackFunc = new MyCamera.cbOutputdelegate (cbExceptiondelegate);**

   While, the cbExceptiondelegate indicates the callback handling function, and it firstly makes the operations of CloseDevice and DestroyHandle, then continuously attempts to reconnect the camera.
3. **Register callback function by calling the callback function registration API after open the camera.**

   Example: **m_pMyCamera.MV_CC_RegisterExceptionCallBack_NET (pCallBackFunc, IntPtr.Zero);**

Note:

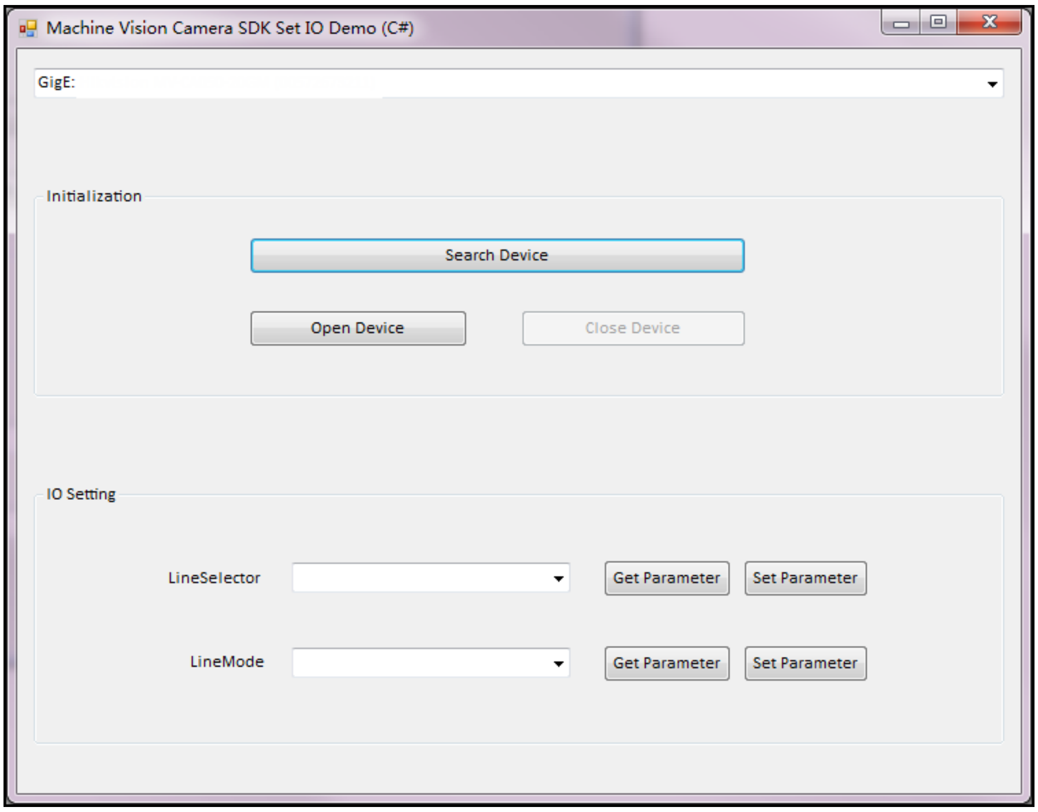
When the camera is disconnected, there will be callback exception, and you can reconnect the camera in exceptional callback.

Chapter 4 SetIODemo

The Demo in this section mainly realizes the control of camera IO input and output.

4.1 Interface Overview

The SetIODemo based on C# language for machine vision camera can realize the function of device search, device control and IO settings.



4.2 Operation Procedure

The operation procedure of SetIODemo is similar with that of BasicDemo, please refer to *Chapter 2.2 Operation Procedure* for details.
After opening a device, you can get and set the camera IO properties, e.g., LineSelector and LineMode. Click Get Parameter or Set Parameter to read or write the corresponding property.

4.3 Programming Guideline

4.3.1 IO Property

There are two IO properties, LineSelector or LineMode.
LineSelector: Select IO port. Three IO ports are available: Line0, Line1 and Line2. Line0 – Can be configured as input only; Line1 – Can be configured as output only; Line2 – Can be configured input or output.
LineMode: Input or output mode.

4.3.2 APIs for Getting and Setting

In the sample program, the APIs used to get and set IO are m_pMyCamera.MV_CC_GetEnumValue_NET(string strKey, ref CSI.MVCC_ENUMVALUE pstValue), and m_pMyCamera.MV_CC_SetEnumValue_ NET (string strKey, UInt32 nValue).
In the SDK, the API function which is similar with the format of *Set* or *Get + Data Type +Value* is a general API used to get or set any camera properties. The first parameter in the general API is property name, which is a *string* type string and can be found in the Feature Tree of MVS, while the second parameter is the obtained or configured property value.

4.3.3 IO Operation

The type of property nodes <LineSelector> and <LineMode> in the Demo is *Enumeration* type. Call general API can realize the property operations.

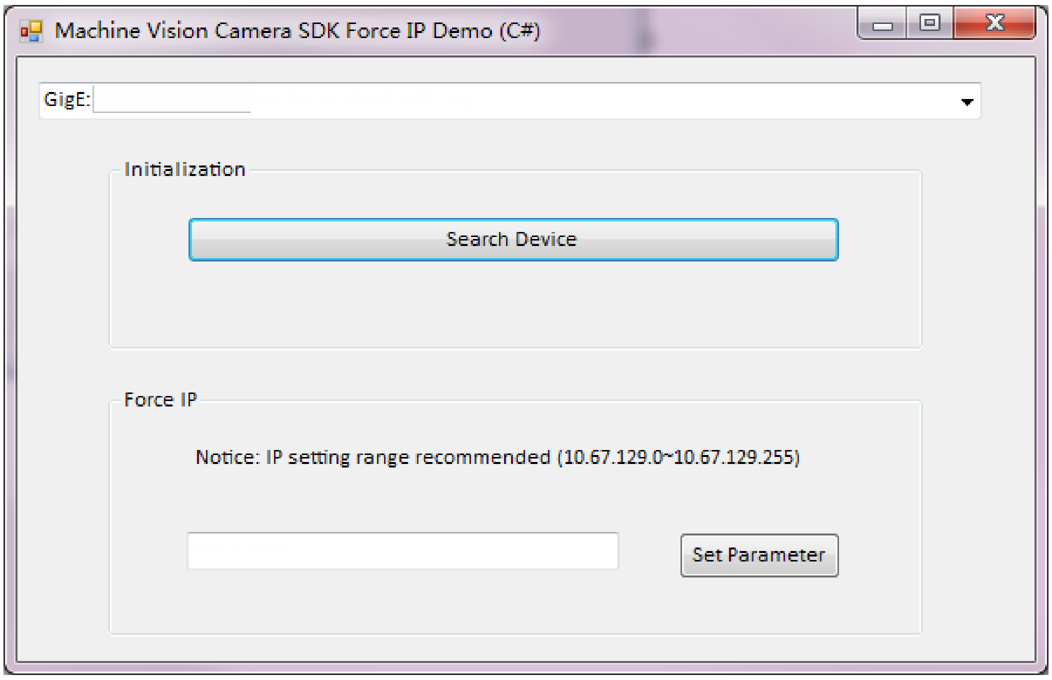
Get:
```
MyCamera.MVCC_ENUMVALUE stSelValue =
            new MyCamera.MVCC_ENUMVALUE();
            nRet = m_pMyCamera.GetEnumValue("LineSelector", ref stSelValue);
MyCamera.MVCC_ENUMVALUE stModeValue =
            new MyCamera.MVCC_ENUMVALUE();
            nRet = m_pMyCamera.GetEnumValue("LineMode", ref stModeValue);
```

Set:
```
nRet = m_pMyCamera.SetEnumValue("LineSelector", nValue);
nRet = m_pMyCamera.SetEnumValue("LineMode", nValue);
```

Chapter 5 ForceIPDemo

5.1 Interface Overview

The ForceIPDemo based on C# language for machine vision camera can realize the function of device search and IP address settings.



Notice: IP setting range recommended (10.67.129.0~10.67.129.255)

5.2 Operation Procedure

Steps:

1. **Click Search Device to enumerate the devices in the IP segment.**

   Note: **The first device item in the searched list will be selected automatically.**
2. **Select a device to configure IP address.**
3. **Input desired IP address in the text field.**

   Note: **In the Set IP field, the IP segment of local NIC and suggested IP range will be display in prompt information.**
4. **Click Set Parameter to set the IP address.**

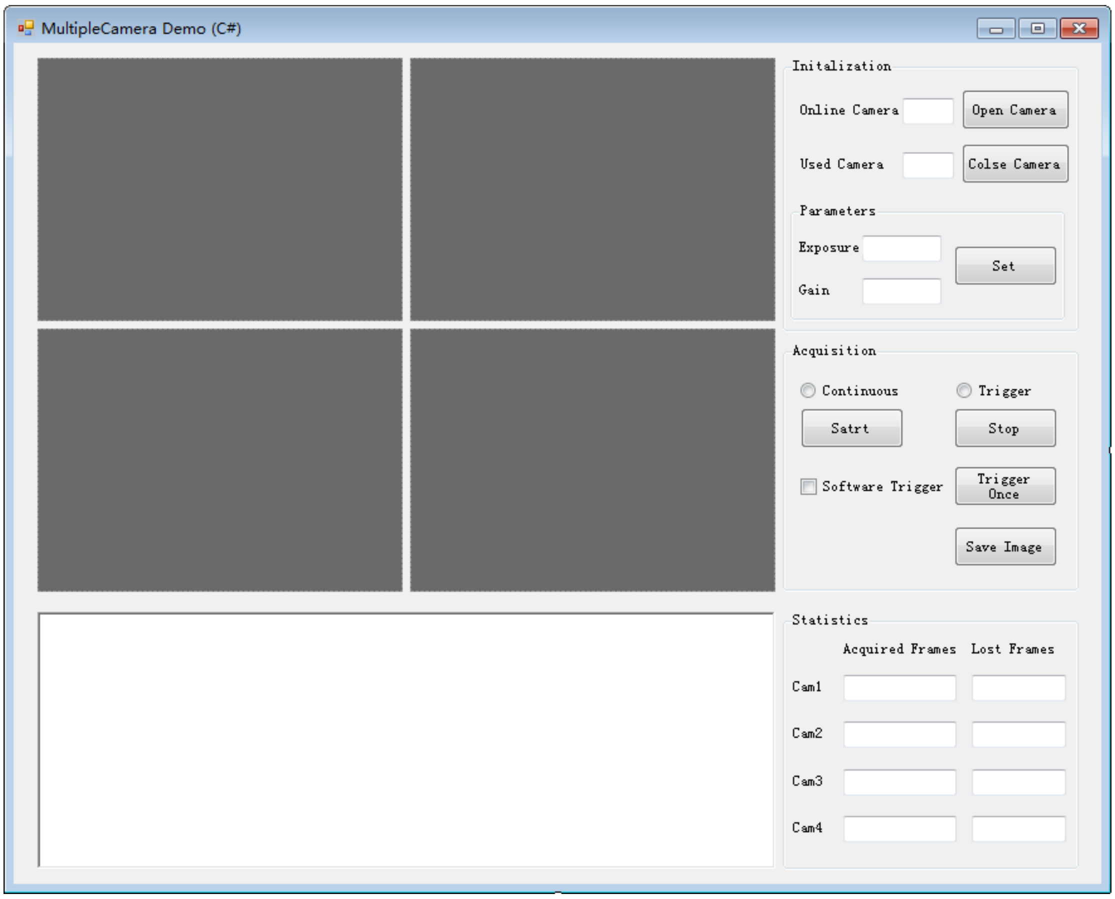
5.3 Programming Guideline

After setting the IP address, call API MyCamera.MV_GIGE_ForceIp_NET(UInt32 nIp) in the SDK.

Chapter 6 MultipleDemo

6.1 Interface Overview

The MultipleDemo based on C# language for machine vision camera can realize the function of initialization, parameter settings, image acquisition, image display and frame information display.



6.2 Operation Procedure

Steps:

1. **The online device number will be enumerated automatically and display in the Online Device field after opening the Demo.**
2. **Input required camera number in the Used Device field.**
3. **Click Open Camera to open the devices in corresponding number continuously.**
   Note: **After initializing, the Parameter field and Image Acquisition field will be active.**
4. **Input parameters in Exposure and Gain field to edit.**
5. **Click Set Parameter to edit the corresponding parameters of all opened devices.**
6. **Select image acquisition mode as continuous or trigger mode.**
7. **Click Start to start the acquisition.**

The live image will display in the left display area.

The acquired frame number and lost frame number will be refreshed (refresh per second).

8. **Click Save to save the four camera's pictures as files named by *image1-image4.bmp* under the directory of executing program.**
9. **(Optional) Click Stop and Close to end the operation.**

Note: **If exception or error occurred during the procedure, the message will be output.**

6.3 Programming Guideline

6.3.1 Multiple Cameras

Based on the BasicDemo, the MultipleDemo added a member variable *m_bEnabled* array in the class. This variable is used to enable four cameras, and the *m_bEnabled* status (True or False) is determined by the used device number and opened device number when initializing. The following basic operations (do or not) of corresponding cameras are also determined by *m_bEnabled*.

6.3.2 Total/Lost Frame Number and Picture Storage

The total frame number (member variable) is calculated in the callback function.

Get the lost frame number by calling API CSI.MV_CC_GetAllMatchInfo_CSI (ref pstInfo).

The update period of total frame and lost frame is 1 second. Set the timer and get the lost frame number per second, and then update the total frame number and lost frame number.

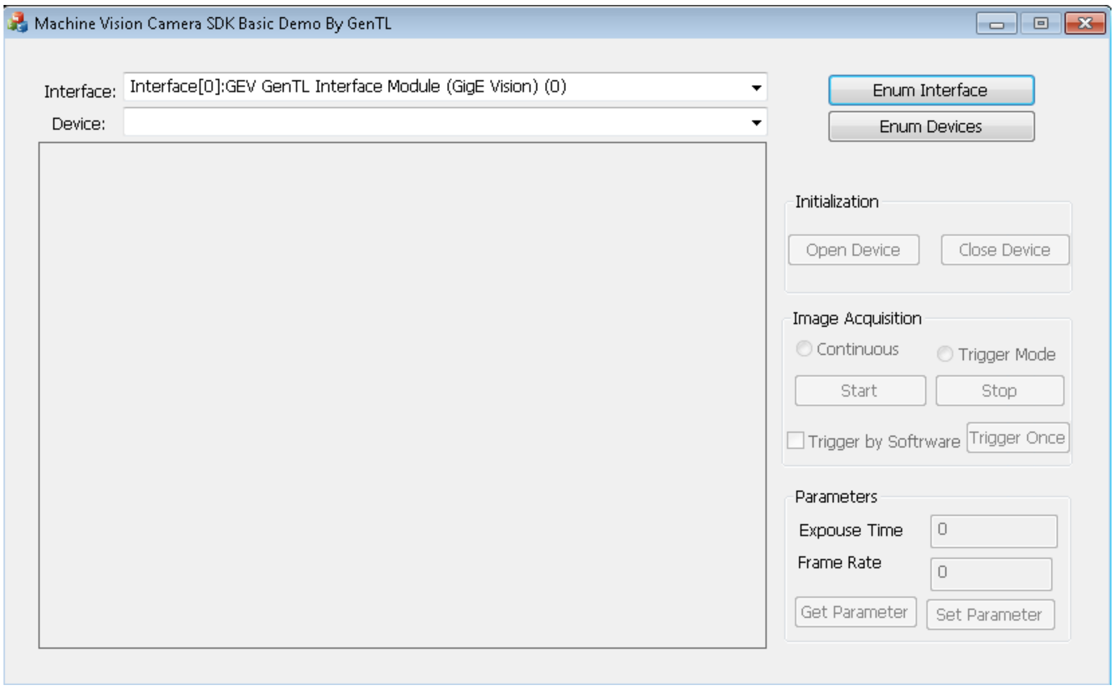
The callback function can also save the pictures and judge whether to save the current frame as picture via the click of Save Picture button. After saving the picture, edit the flag bit to avoid saving picture repeatedly.

Chapter 7 BasedOnGenTL

The BasedOnGenTL is a sample program based on GenTL, it can load different CTI files for enumerating corresponding devices, and realizing the functions of interface enumeration, device search, initialization, image acquisition, and parameters control.

7.1 Interface Overview

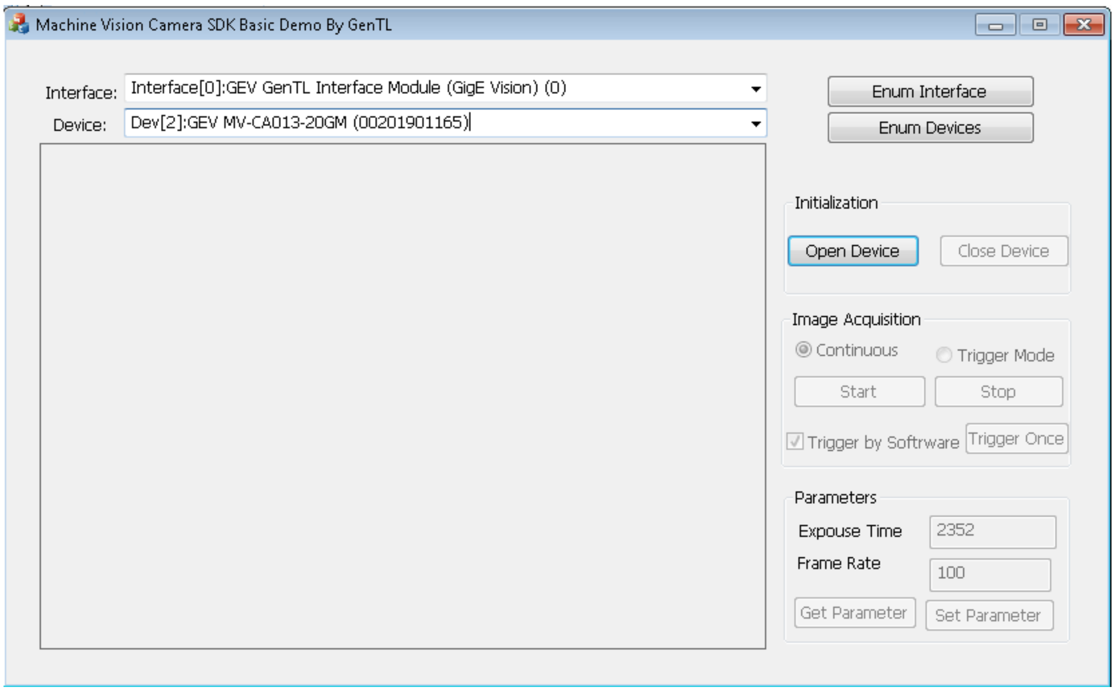
The BasedOnGenTL interface includes five control modules (interface enumeration, device search, initialization, image acquisition, and parameters control), one device drop-down list, and one image display area.



7.2 Operation Procedure

Steps:

1. Click **Enumerate Interface** to browse repository.
2. Select required CTI file to load.
   For example, select the MvProducerGEV.cti file in the directory: *C:\Program Files (x86)\Common Files\MVS\Runtime\Win32_i86*.
3. Click **Search Device** to search the online device.
   The online GigeVision devices appear in the drop-down list of Device field.
   Note: If the user ID is not empty, the devices will be displayed as “device type” + “device name” + “IP address”; otherwise the devices will be displayed as “device type” + “device model” + ”IP address”.

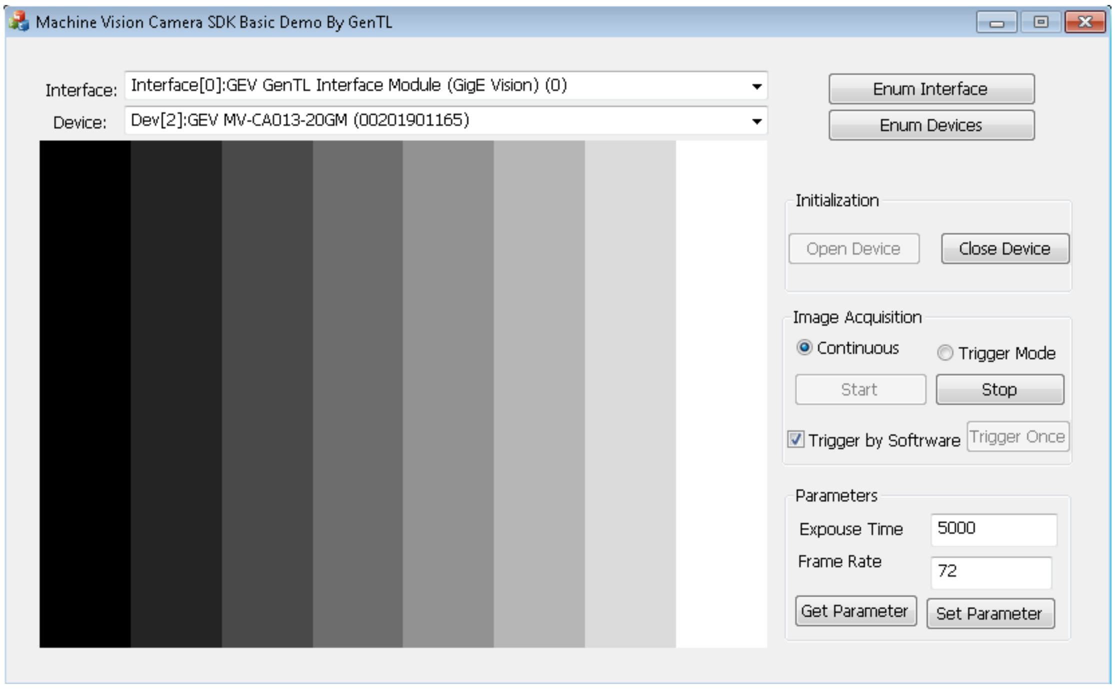


Interface: Interface[0]:GEV GenTL Interface Module (GigE Vision) (0)
Device: Dev[2]:GEV MV-CA013-20GM (00201901165)
Expouse Time 2352
Frame Rate 100

4. **Click to select a device in the drop-down list of Device field.**
5. **Click Open Device button in the Initialization field to active the Image Acquisition field.**
6. **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.**

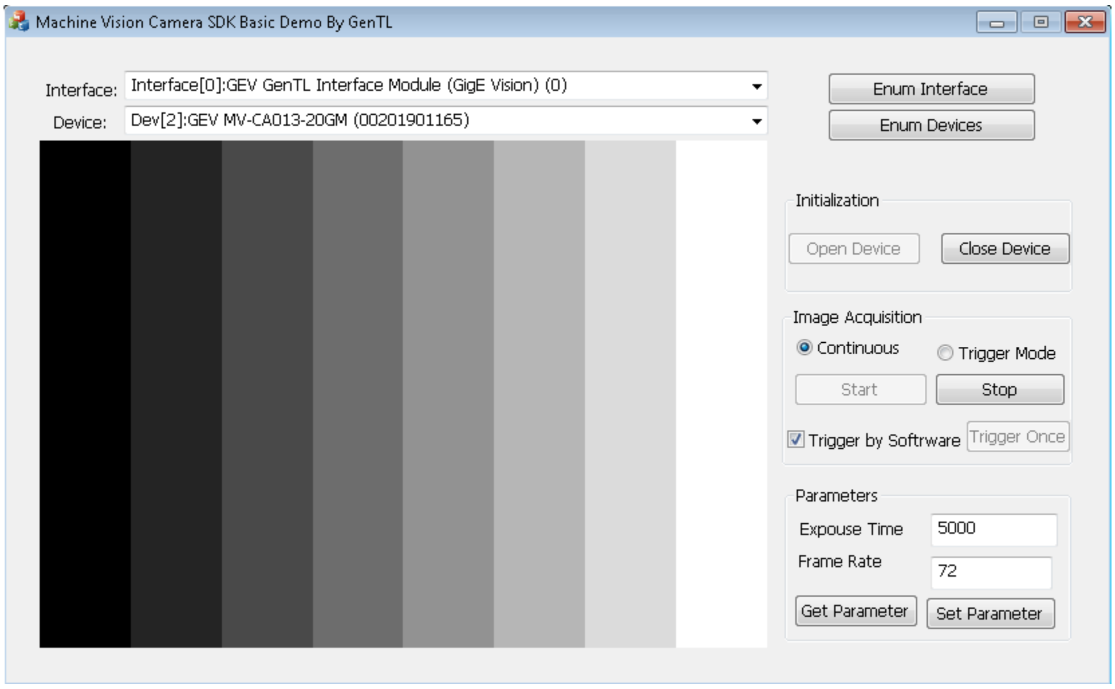


Interface: Interface[0]:GEV GenTL Interface Module (GigE Vision) (0)
Device: Dev[2]:GEV MV-CA013-20GM (00201901165)
Expouse Time 5000
Frame Rate 72

7. **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.



Expouse Time: 5000

Frame Rate: 72

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

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

7.3 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.
2. Configure project.
   3) Create CS project.
   4) Add *MvCameraControl.Net.dll* to the project.
3. Reference the naming space *using MvCamCtrl.NET* in the project to call the camera operation function of *My Camera*.