Machine Vision Camera SDK Demo (VC60)

User Manual

User Manual

About this Manual

This Manual is applicable to Machine Vision Camera SDK Demo (VC60).

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 C++ language.

Thirteen VC6.0 Demos are provided in the SDK directory, including five MFC Demos and eight console Demos.

MFC Demos: Basic Demo, Reconnect Demo, Set IO Demo, Force IP Demo, and Multiple Camera.

Console Demos: ConnectSpecCamera, ConvertPixelType , Events , Grab_Callback, GrabImage, MultiCast, ParametrizeCamera_FileAccess, and ParametrizeCamera_LoadAndSave.

All the Demos introduce the API calling methods of machine vision camera.

To ensure the proper use of SDK, please refer to the contents below and read the manual carefully before operation and development.

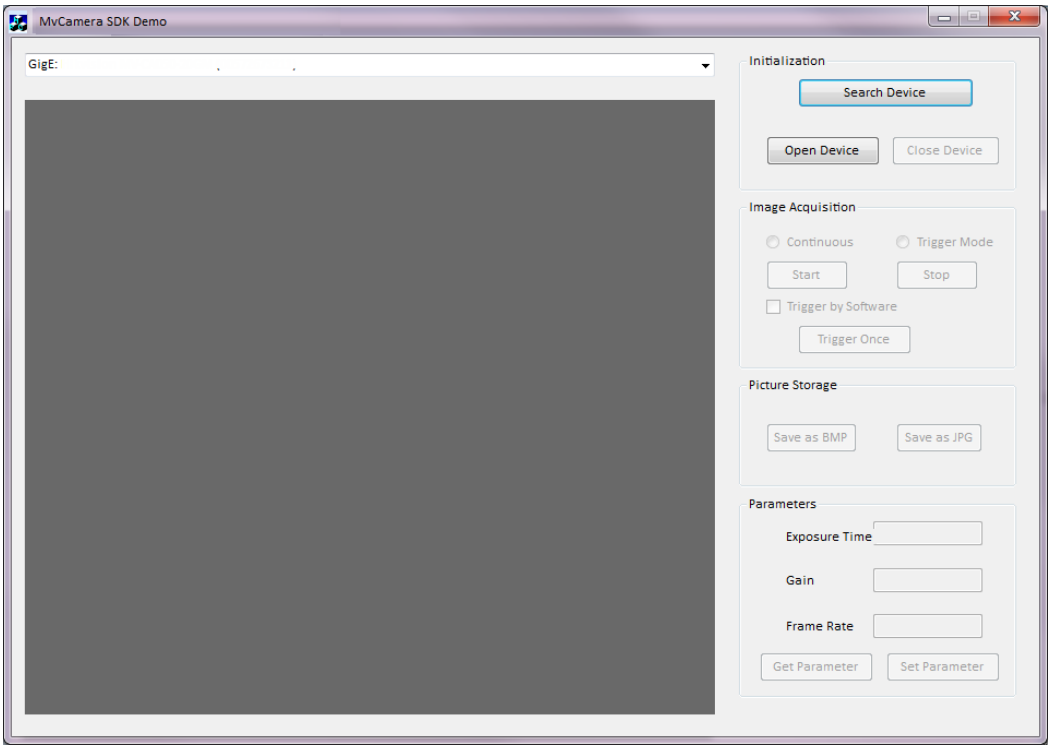
Chapter 2 Basic Demo

Basic Demo 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 Basic Demo, as it contains multiple required examples.

2.1 Interface Overview

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



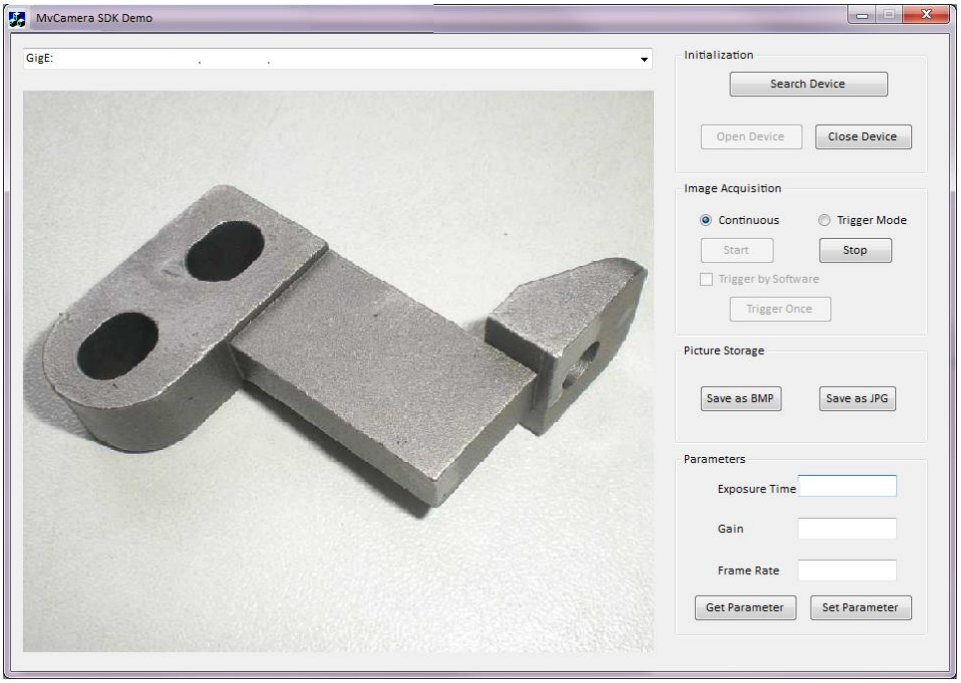
2.2 Operation Procedure

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 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 VS project.
   2) Add *MvSdkExport.lib* and *MvSdkExport.h* to the project.
3. Reference the library file and header file to call the camera operation function of *MvSdkExport.h*.

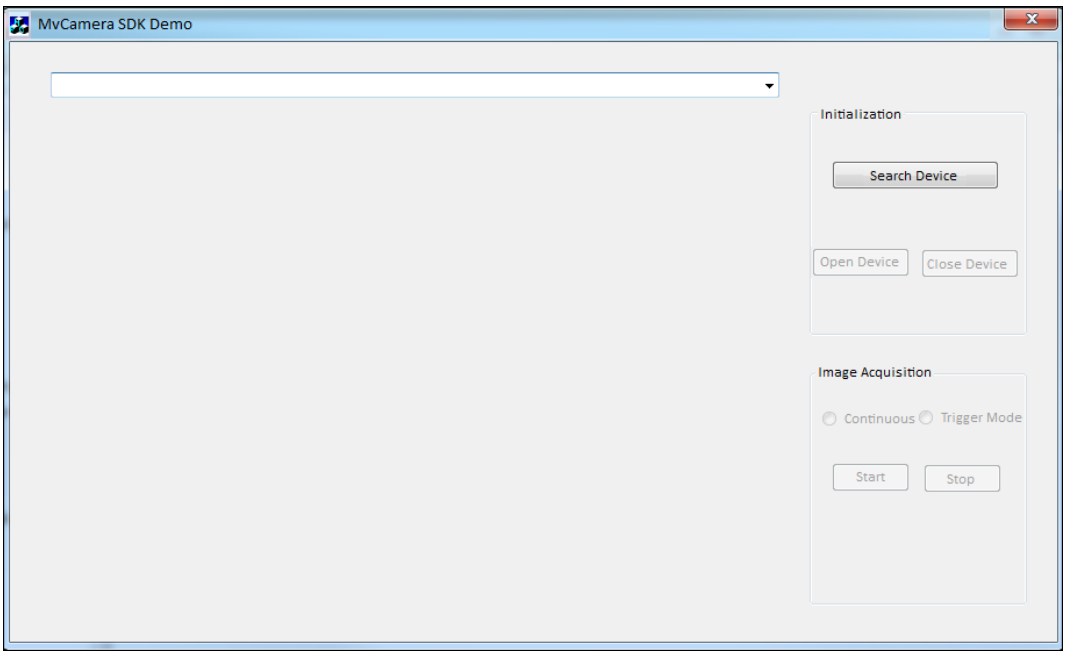
Chapter 3 Reconnect Demo

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

When the camera is disconnected, there will be callback exception, and the Reconnect Demo will attempt to connect the camera according to the selected camera information. And the online cameras will be connected.

3.3 Programming Guideline

The programming guidance of Reconnection Demo is similar with that of Basic 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 exceptional disconnection callback is *RegisterExceptionCallBack* in the machine vision camera SDK (C++).

Steps:

1. Create the reconnection function *ReconnectDevice* in *CBasicDemoDlg* class.
2. Transmit to callback function *RegisterExceptionCallBack*.
3. Register callback function by calling the callback function registration API after opening the camera.

   Example: m_pcMyCamera->RegisterExceptionCallBack(ReconnectDevice, this);

Note:

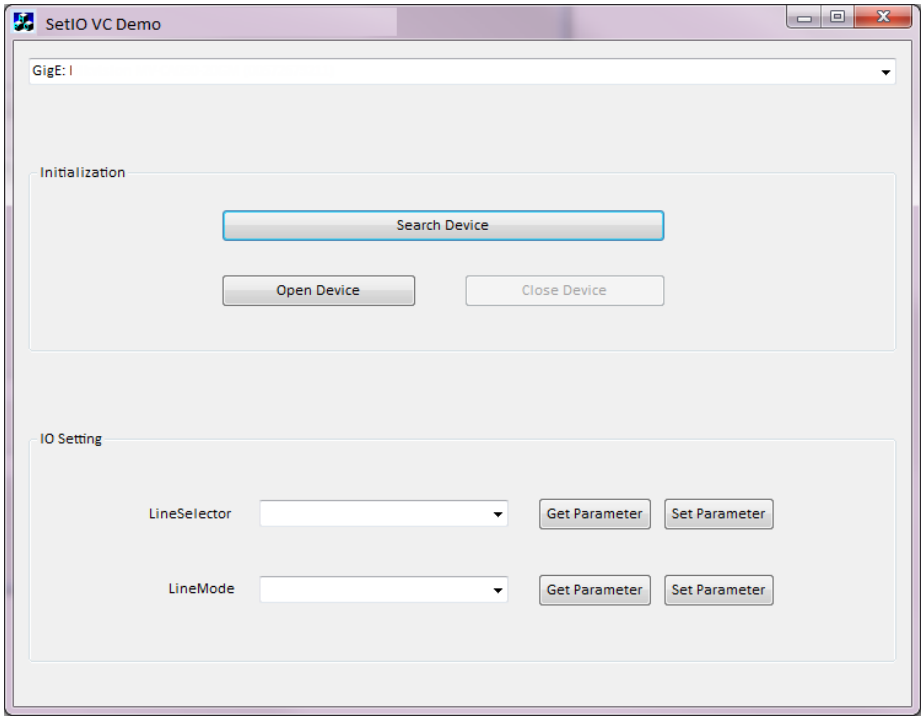
When the camera is disconnected, there will be callback exception, and you can reconnect the camera in exceptional callback.

Chapter 4 Set IO Demo

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

4.1 Interface Overview

The Set IO Demo 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 Set IO Demo is similar with that of Basic Demo, 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 Demo, the APIs used to get and set IO are MV_CC_GetEnumValue(IN void* handle,IN const char* strKey,OUT MVCC_ENUMVALUE *pEnumValue) and MV_CC_SetEnumValue(IN void* handle,IN const char* strKey,IN unsigned int 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 searched in the file *MvCameraNode.xls*, 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:

```
nRet = m_pcMyCamera->GetEnumValue("LineSelector", &stSelector);
nRet = m_pcMyCamera->GetEnumValue("LineMode", &stSelector);
```

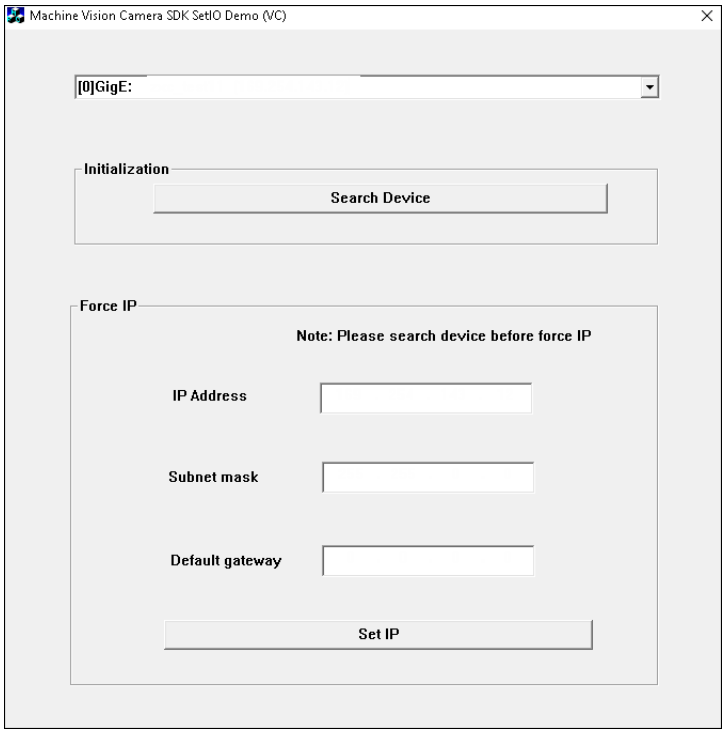
Set:

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

Chapter 5 Force IP Demo

5.1 Interface Overview

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



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 displayed in prompt information.
4. Click **Set Parameter** to set the IP address.

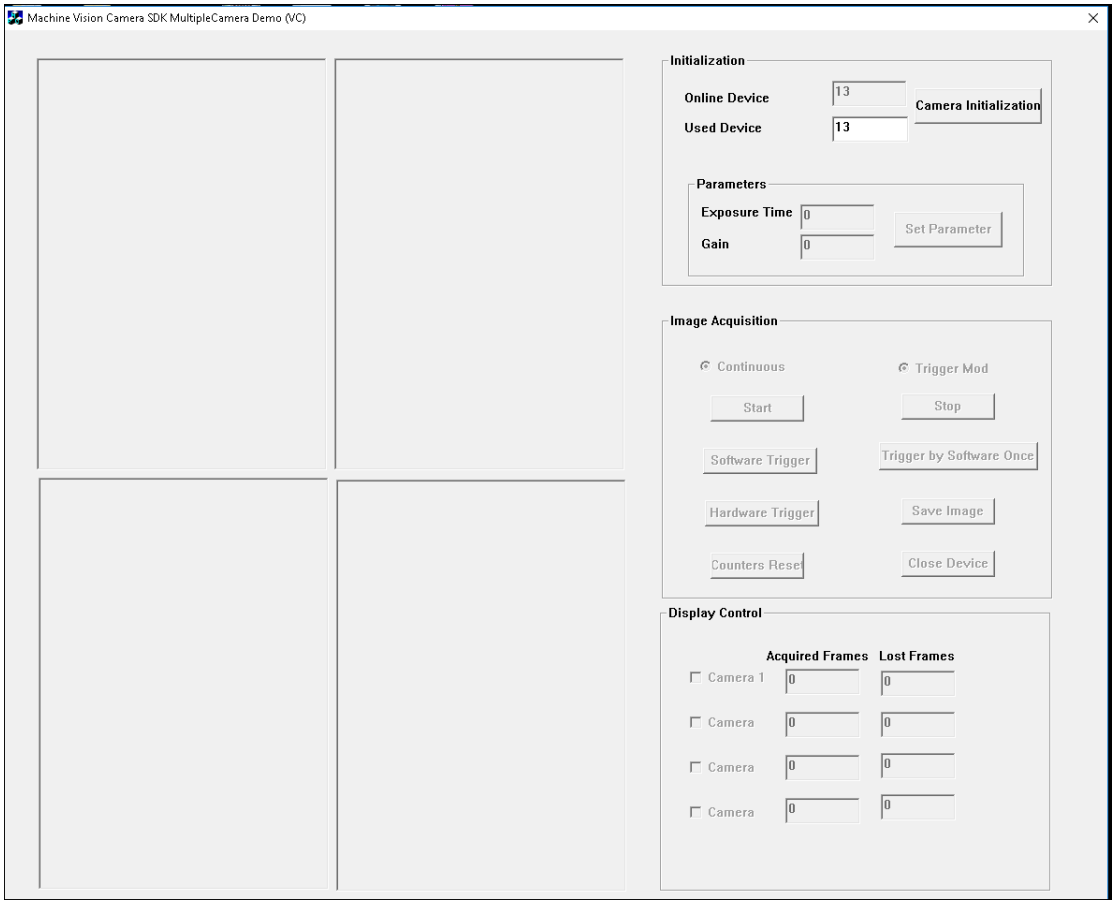
5.3 Programming Guideline

After setting the IP address, call API MV_GIGE_ForceIpEx(IN void* handle, unsigned int nIP, unsigned int nSubNetMask, unsigned int nDefaultGateWay) in the SDK.

Chapter 6 Multiple Demo

6.1 Interface Overview

The Multiple Demo 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 **Camera Initialization** 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 cameras' 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 prompt dialog will pop up.

6.3 Programming Guideline

6.3.1 Multiple Cameras

Based on the Basic Demo, the Multiple Demo added a member variable *m_nUseNumEdit* array in the class, which indicates serial number of four cameras. The following basic operations (do or not) of corresponding cameras are determined by the used device number and opened device serial number when initializing.

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 m_pcMyCamera[nUsingDeviceNum]->

GetAllMatchInfo(&struMatchInfo).

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.