Machine Vision Camera SDK Demo (LabVIEW)

User Manual

User Manual

About this Manual

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

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 LabVIEW.

Two LabVIEW sample programs, including single camera sample Samples and two cameras sample TwoCameraSamples, are provided in the SDK directory. The Demos are developed by adopting MvCameraControlSDK. In addition, the SDK C language version is also encapsulated into LabVIEW VI, which supports calling sub VI by LabVIEW.

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

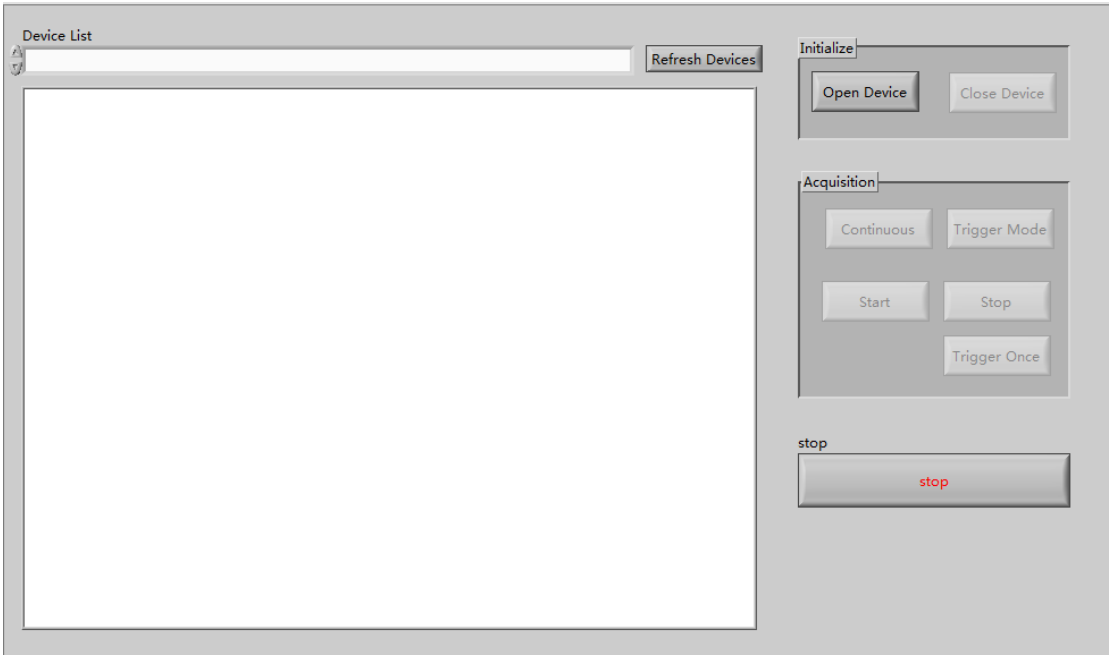
Chapter 2 Samples

Samples 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 Samples, as it contains multiple required examples.

2.1 Interface Overview

The Samples for machine vision camera can realize the function of device refresh, initialization, and image acquisition.

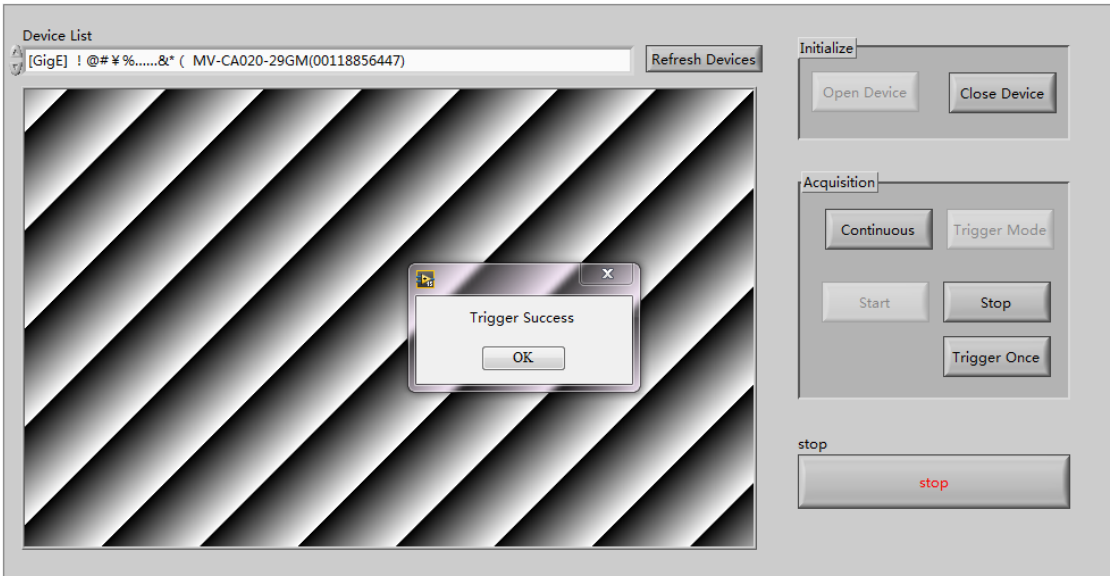


2.2 Operation Procedure

Steps:

1. Click **Refrsh Devices** 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 name”; when the device name is empty, the devices will be displayed as “device model” + ”Serial No.”.
2. Click to select a device in the drop-down list.
3. Click **Open Device** button in the Initialization field to active the Acquisition field.
4. Select image acquisition mode as **Continuous** or **Trigger Mode**.
   Notes:
   The default image acquisition mode is **Continuous**.
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.



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

2.3 Programming Guideline

2.3.1 Introduction of LabVIEW VI

To call the SDK of C API in the LabVIEW system conveniently, sub VI files are encapsulated into the LabVIEW system. There are total 20 sub VI files in the *MvCameraLib.lvlib*: EnumDevices.vi, CreateHandle.vi, DestroyHandle.vi, OpenDevice.vi, CloseDevice.vi, StartGrabbing.vi and StopGrabbing.vi are used for camera operations, SetValue.vi and GetValue.vi are used to set and get camera parameters, SaveImage.vi is used to save pictures. This VI files are the secondary encapsulation of SDK of C API, so they contain serval DLLs in the SDK of C API.

Here we take EnumDevices.vi as an example. The input parameters of this API are nLayerType and error input. nLayerType is a 32-bit integer, and it is used to input the enumerated device types, the value 1 represents camera with GigE port, while the value 4 represents camera with U3V port. The output parameters contain function return, DeviceNum, pstGigEDevArray, pstU3VDevArray, and error output. Function return is the returned value of called API, return 0 for success, and return negative number (corresponds to error code) for failure. DeviceNum is the number of enumerated cameras. pstGigEDevArray is the cluster array of GigE camera information, while pstU3VDevArray is the cluster array of USB3 vision camera information. See the definition of device information structure below:

```
typedef struct _MV_CC_DEVICE_INFO_
{
    // common info
    unsigned short      nMajorVer;
    unsigned short      nMinorVer;
    unsigned int        nMacAddrHigh;    // MAC address
```

```
    unsigned int        nMacAddrLow;
    unsigned int        nTLayerType;    // Device transport layer protocol type, e.g.,
MV_GIGE_DEVICE
    unsigned int        nReserved[4];
    union
    {
        MV_GIGE_DEVICE_INFO stGigEInfo;
        MV_USB3_DEVICE_INFO stUsb3VInfo;
        // more ...
    }SpecialInfo;

}MV_CC_DEVICE_INFO;
```

2.1.1 LabVIEW Demo Development

Steps:

1. Create a project in the LabVIEW.
2. Right-click My Computer in the project manager to select files.
3. Select MvCameraLib.lvlib to add it to the project.
4. Create a VI and name it as *DisplayPanel.vi*.
5. Add vi files in the MvCameraLib.lvlib to the programming panel for camera operations.

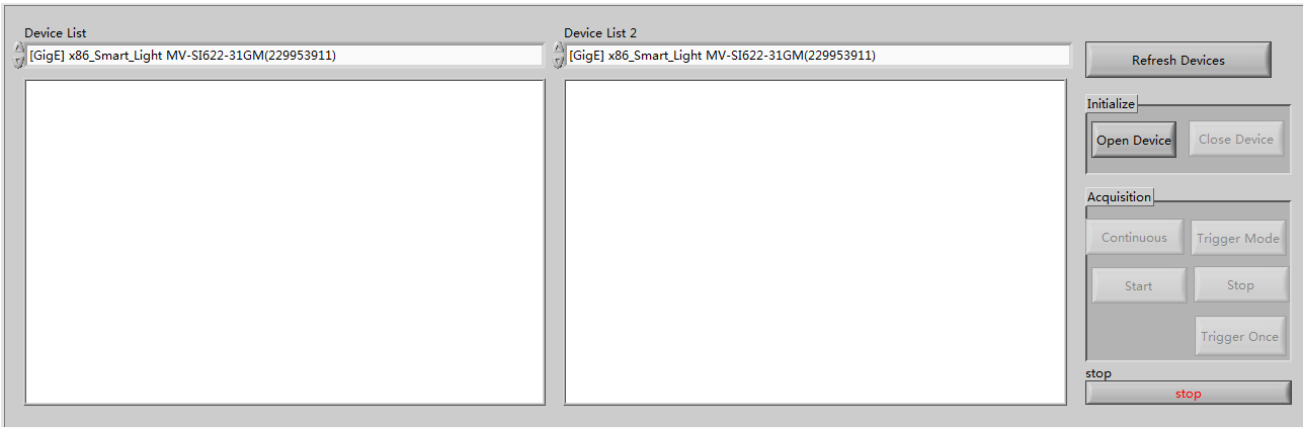
Note: When LabVIEW is loading and initializing *vi* files, the progress dialog will open. This dialog will search *.vi* files automatically and call them to *MvCameraControl.dll* and *MvCameraPatch.dll*. If searching failed, you should add the path of *MvCameraControl.dll* and *MvCameraPatch.dll* in the SDK manually.

Chapter 3 TwoCameraSamples

TwoCameraSamples is a sample program of LabVIEW two cameras, which can realize the function of two cameras control.

3.1 Interface Overview

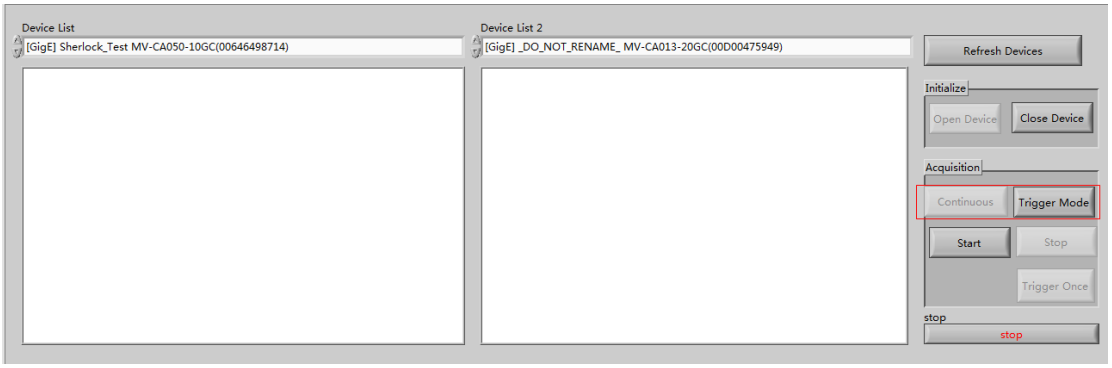
The TwoCameraSamples for machine vision camera can realize the function of device refresh, initialization, and image acquisition.



3.2 Operation Procedure

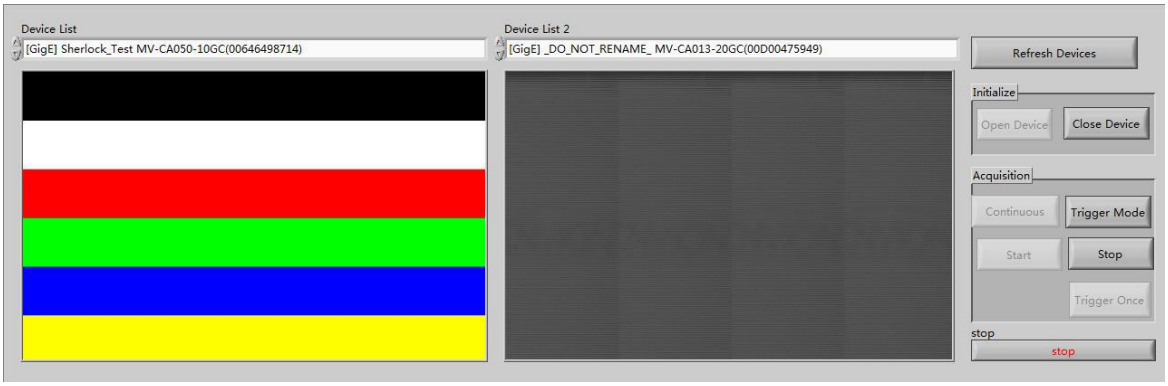
Steps:

1. Click **Refrsh Devices** 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 name”; when the device name is empty, the devices will be displayed as “device model” + ”Serial No.”.
2. Click to select a device in the drop-down list.
3. Click **Open Device** button in the Initialization field to active the Acquisition field.



4. Select image acquisition mode as **Continuous** or **Trigger Mode**.
   Notes:
   The default image acquisition mode is **Continuous**.

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.



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

3.3 Programming Guideline

3.3.1 Introduction of LabVIEW VI

To call the SDK of C API in the LabVIEW system conveniently, sub VI files are encapsulated into the LabVIEW system. There are total 20 sub VI files in the *MvCameraLib.lvlib*: EnumDevices.vi, CreateHandle.vi, DestroyHandle.vi, OpenDevice.vi, CloseDevice.vi, StartGrabbing.vi and StopGrabbing.vi are used for camera operations, SetValue.vi and GetValue.vi are used to set and get camera parameters, SaveImage.vi is used to save pictures. This VI files are the secondary encapsulation of SDK of C API, so they contain serval DLLs in the SDK of C API.

Here we take EnumDevices.vi as an example. The input parameters of this API are nLayerType and error input. nLayerType is a 32-bit integer, and it is used to input the enumerated device types, the value 1 represents camera with GigE port, while the value 4 represents camera with U3V port. The output parameters contain function return, DeviceNum, pstGigEDevArray, pstU3VDevArray, and error output. Function return is the returned value of called API, return 0 for success, and return negative number (corresponds to error code) for failure. DeviceNum is the number of enumerated cameras. pstGigEDevArray is the cluster array of GigE camera information, while pstU3VDevArray is the cluster array of USB3 vision camera information. See the definition of device information structure below:

```
typedef struct _MV_CC_DEVICE_INFO_
{
    // common info
    unsigned short      nMajorVer;
    unsigned short      nMinorVer;
    unsigned int        nMacAddrHigh;    // MAC address
    unsigned int        nMacAddrLow;
```

```
    unsigned int        nTLayerType;    // Device transport layer protocol type, e.g.,
MV_GIGE_DEVICE
    unsigned int        nReserved[4];
    union
    {
        MV_GIGE_DEVICE_INFO stGigEInfo;
        MV_USB3_DEVICE_INFO stUsb3VInfo;
        // more ...
    }SpecialInfo;

}MV_CC_DEVICE_INFO;
```

3.3.2 LabVIEW Demo Development

Steps:

1. Create a project in the LabVIEW.
2. Right-click My Computer in the project manager to select files.
3. Select MvCameraLib.lvlib to add it to the project.
4. Create a VI and name it as *TwoCameraDisplayPanel.vi*.
5. Add vi files in the MvCameraLib.lvlib to the programming panel for camera operations.

Note: When LabVIEW is loading and initializing *vi* files, the progress dialog will open. This dialog will search *.vi* files automatically and call them to *MvCameraControl.dll* and *MvCameraPatch.dll*. If searching failed, you should add the path of *MvCameraControl.dll* and *MvCameraPatch.dll* in the SDK manually.