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.

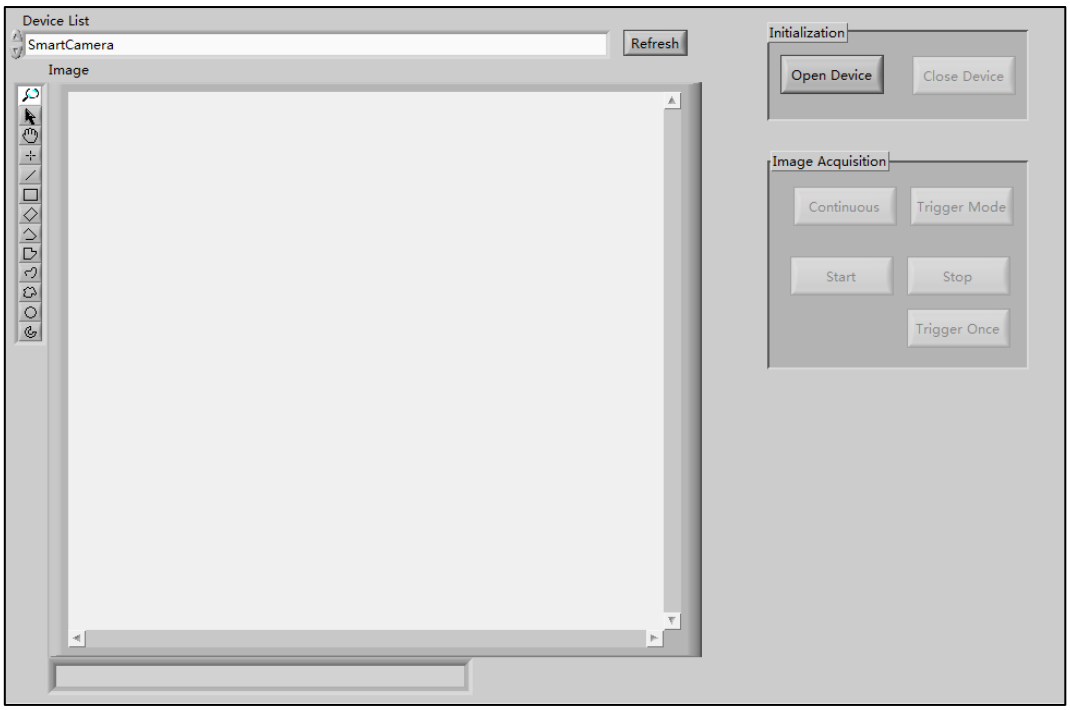
One LabVIEW sample program, e.g., BasicDemo, is provided in the SDK directory. This Demo is developed by adopting MvCameraControlSDK. In addition, the SDK C language version is also encapsulated into LabVIEW VI, which supports calling sub VI by LabVIEW.

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

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

Note:

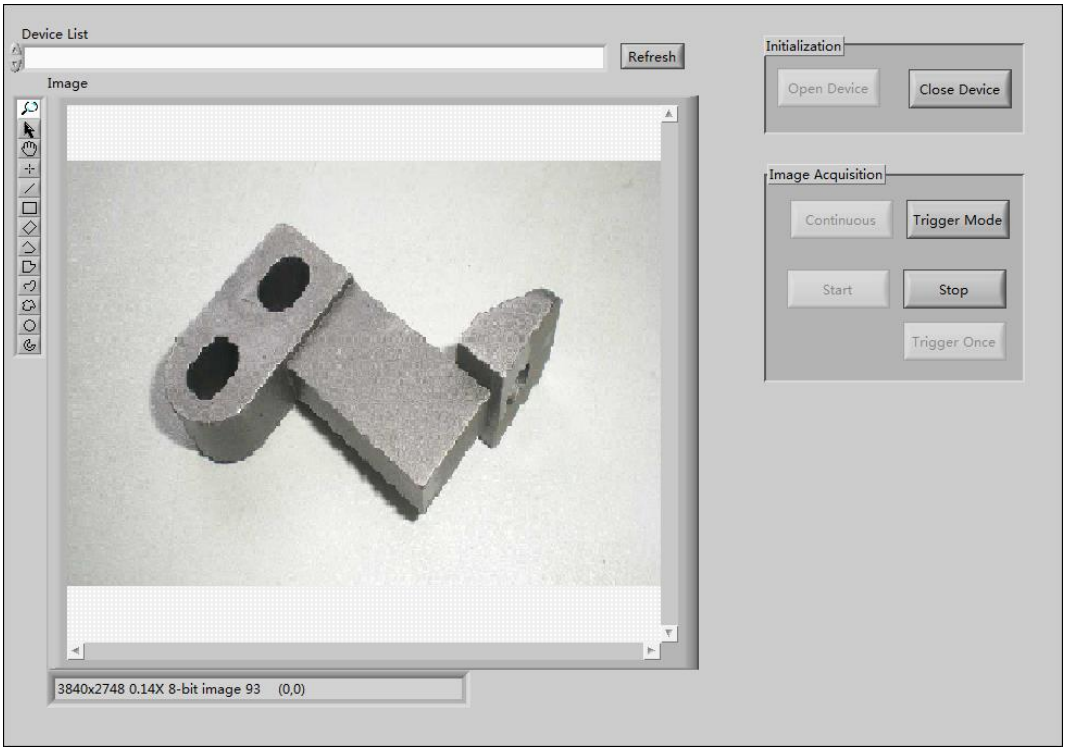
Now the Basic Demo based on LabVIEW only supports Mono8 format.



Chapter 2 Operation Procedure

Steps

1. Click **Refresh** in the Device List field to search the online device.
   The online devices will display in the drop-down list in the Device List field.
   Note: If the user ID is not empty, the device name will display.
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. 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.

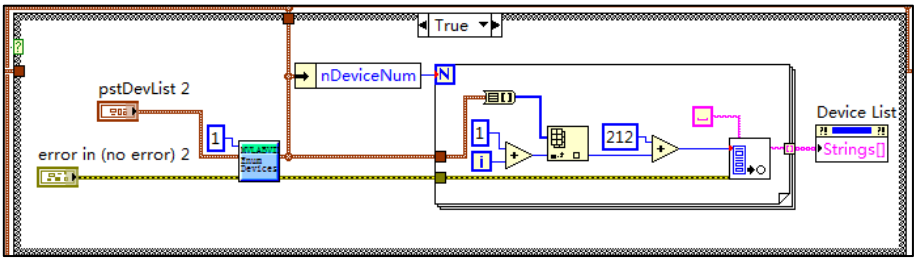
Note: If exception or error occurred during the procedure, the prompt dialog will pop up.

Chapter 3 Programming Guideline

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, pstDevList 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. pstDevList is with cluster structure, which is used to output, the output parameters contain function return, pstDevList 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. The device list information will be returned by pstDevList, the first member in the cluster represents the existing device number in the device list, the following members (from No.2 to No.257) represent the pointer for device information structure, each pointer address will store the information of one device. Adopt the following structure to get the detailed information of each device structure:



- Unit 1 transforms the cluster to arrays for easy handling.
- Unit 2 indexes the arrays to get the values in the arrays according to index No., and a *For* loop structure will be used to get the pointer address of device information structure one by one. 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;
```

- Unit 3 gets the offset of variables in the structure. For example, the offset of variable *nMajorVer* is 0, the offset of variable *nMinorVer* is 2, and so on. The offset 212 in the figure represents the offset of device UserID.
- Unit 4 is a VI file (GetValueByPointer.vi) provided by LabVIEW, and it is in the path of *<LabVIEW>\vi.lib\Utility\imports\GetValueByPointer\GetValueByPointer.xnode*. This VI file is an API used for pointer operation, and you can get the value of address pointed by the pointer via this function.

3.2 Developing LabVIEW Demo

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 *BasicDemo.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 following dialog will pop up. This dialog will search *.vi* files automatically and call them to *MvCameraControl.dll*. If searching failed, you should add the path of *MvCameraControl.dll* in the SDK manually.