OpenCV

API Document

Version 1.0

e-con Systems

4/6/2018

e-con Systems

Your Product Development Partner

Disclaimer

e-con Systems reserves the right to edit/modify this document without any prior intimation of whatsoever.

Contents

Introduction to OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV libraries are used to communicate with cameras. APIs introduced in the OpenCV can be supported with all e-con Systems cameras.

This document provides the details about the APIs introduced in the OpenCV Video Capture class.

Description

The APIs introduced in the OpenCV can be used to communicate with the cameras. Human Interface Device(HID) related APIs were implemented in the OpenCV, which will be useful to access HID device information of the camera. The APIs introduced were solely related to communicate with camera properties and HID settings of the camera.

Introduced APIs

The details regarding the APIs introduced in OpenCV are covered in this section.

BOOL getDevices(int &devices)

This function retrieves the number of camera devices connected to the PC.

Parameters	Description	Return Values
`int &devices`	Number of camera devices connected to the port is retrieved.	TRUE on Success FALSE on Failure Number of Devices Connected

Sample Code

```
VideoCapture cap;
if(cap.getDevices(devices))
{
      cout << "The number of camera devices connected to the port is " << devices << '\n';
}
```

BOOL getDeviceInfo(int index, String &deviceName, String &vid, String &pid, String &devicePath)

This function will retrieve the Name, Vendor ID, Product ID and Path of the camera device connected to the port. Before calling this API, you must know the index of the camera device connected to the port.

Parameters	Description	Return Values
`int index`	You must input the camera device index.	TRUE on Success FALSE on Failure
`String &deviceName`	Retrieves the Name of the camera device.	
`String &vid`	Retrieves the Vendor ID of the camera device.	
`String &pid`	Retrieves the Product ID of the camera device.	
`String &devicePath`	Retrieves the Path of the camera device.	

Sample Code

```
VideoCapture cap;
```

```
if(cap.getDevices(devices))
{
      for(int i = 0; i < devices; i++)
      {
            if(cap.getDeviceInfo(i, deviceName, vid, pid,
devicePath))
            {
                  cout << "Camera Device Name = " <<
deviceName << endl;
                  cout << "Vendor ID = " << vid << endl;
                  cout << "Product ID = " << pid << endl;
                  cout << "Camera Device Path = " <<
devicePath << endl;
            }
      }
}
```

BOOL getFormats(int &formats)

This function is used to get the total number of Video formats supported by the camera with respect to frames per second (fps). Before calling this API, the camera must be opened.

Parameters	Description	Return Values
`int &formats`	Retrieves the number of formats supported by the camera with respect to fps.	TRUE on Success FALSE on Failure Number of Video formats supported.

Sample Code

```
VideoCapture cap;
cap.open(0);
if(cap.getFormats(formats))
{
      cout << "The total number of formats supported by
the Camera = " << formats << endl;
}
```

BOOL getFormatType(int formats, String &formatType, int &width, int &height, int &fps)

This function is used to get the Format type, width, height and fps of the camera device connected with respect to the index passed. Before calling this API, you must open the camera and pass the input parameter index to this API.

Parameters	Description	Return Values
`int formats`	Integer value to be passed as index to the API.	TRUE on Success FALSE on Failure
`String &formatType`	Retrieves the Format type of the camera with respect to the index passed.	
`int &width`	Retrieves the width of the resolution with respect to the index passed.	
`int &height`	Retrieves the height of the resolution with respect to the index passed.	
`int &fps`	Retrieves the fps with respect to the index passed.	

Windows Sample Code

The even index is passed in the getFormatType API, since there are two Video Formats of the same type and resolutions will be distinguished as VIDEOINFOHEADER and VIDEOINFOHEADER2. So, you must use VIDEOINFOHEADER that is, even index.

```
VideoCapture cap;
cap.open(0);
if(cap.getFormats(&formats))
{
      for (int i = 0; i < formats; i++)
      {
            if(i%2 == 0)
            {
                  if(cap.getFormatType(i, formatType,
width, height, fps))
                  {
                        cout << "Video Format = " <<
formatType << endl;
                        cout << "Width = " << vid << endl;
                        cout << "Height = " << pid << endl;
                        cout << "FPS(Frames per Second = "
<< fps << endl;

                  }
```

```
            }
      }
}
```

Linux Sample Code

```
VideoCapture cap;
cap.open(0);
if(cap.getFormats(&formats))
{
      for (int i = 0; i < formats; i++)
      {
            if(cap.getFormatType(i, formatType, width,
height, fps))
            {
                  cout << "Video Format = "<< formatType <<
endl;
                  cout << "Width = " << width << endl;
                  cout << "Height = "<< height << endl;
                  cout << "FPS = " << fps;
            }
      }
}
```

BOOL setFormatType(int index)

This function is used to set the Format type, width, height and fps to the camera. Before calling this API, the camera must be open.

Parameters	Description	Return Values
`int index`	Index to be passed as the input to set the Video Format.	TRUE on Success FALSE on Failure

Sample Code

```
VideoCapture cap;
cap.open(0);
if(cap.getFormats(&formats))
{
      for (int i = 0; i < formats; i++)
      {
            if(cap.getFormatType(i, formatType, width,
height, fps))
            {
```

```
                cout << "Video Format = "<< formatType <<
endl;
                cout << "Width = " << width << endl;
                cout << "Height = "<< height << endl;
                cout << "FPS = " << fps;
            }
        }
}
cin >> index;
if(cap.setFormatType(index))
{
        cout << "Video Format Type is set" <<  endl;
}
```

BOOL get(int propId, long &min, long &max, long &steppingDelta, long &supportedMode, long &currentValue, long &currentMode, long &defaultValue)

This function is used to retrieve the Minimum, Maximum, Stepping Delta, Supported Mode, Current Value, Current Mode and Default Value of the Camera Video Properties. Before calling this API, the camera must be opened and you must know the property ID of the OpenCV Camera Properties (For example, CV_CAP_PROP_BRIGHTNESS for brightness, CV_CAP_PROP_CONTRAST for contrast and so on).

Parameters	Description	Return Values
`int propId`	Input the property ID of the camera properties to the API.	TRUE on Success FALSE on Failure
`long &min`	Retrieves the Minimum value of the camera property supported.	
`long &max`	Retrieves the Maximum value of the camera property supported.	
`long &steppingDelta`	Retrieves the Stepping Delta of the camera property supported.	
`long &supportedMode`	Retrieves the supported Mode of the camera property (1-Auto, 2-Manual, 3-Auto and Manual).	
`long &currentValue`	Retrieves the current value set in the camera property.	
`long &currentMode`	Retrieves the current Mode set in the Camera.	

`Long &defaultValue`	Default value of the camera is retrieved for the camera property.

Sample Code

```
VideoCapture cap;
cap.open(0);
if(get(CV_CAP_PROP_BRIGHTNESS, min, max, steppingDelta,
supportedMode, currentValue, currentMode, defaultValue))
{
      cout << "Brightness Values: " << endl;
      cout << " Minimum Value = " << min << endl;
      cout << " Maximum Value = " << max << endl;
      cout << "Stepping Delta = " << steppingDelta <<
endl;
      cout << "Supported Mode = "<< supportedMode << endl;
      cout << "Current Value = " << currentValue << endl;
      cout << "Current Mode = " << currentMode << endl;
      cout << "Default Value = " << defaultValue << endl;

}
```

BOOL set(int propId, long value, long mode)

This function is used to set the value and mode to the camera property which is passed as input parameter to the API. Before calling this function, the camera must be opened and you must know to which property the value is passed.

1 - Auto mode.

2 - Manual mode.

3 - Auto and Manual mode.

Parameters	Description	Return Values
`int propId`	You must input the property ID of the camera properties to the API.	TRUE on Success FALSE on Failure
`long value`	The value to be set in the camera property is passed in this parameter.	
`long mode`	This parameter is used to change the mode of the camera property. (If the camera supports only auto mode, you cannot set the different value or mode and if the camera supports only manual mode, you cannot set the auto mode).	

Sample Code

```
VideoCapture cap;
cap.open(0);
if(cap.set(CV_CAP_PROP_BRIGHTNESS, 12, 2))
{
      cout << "Camera Brightness value is set" << endl;
}
```

BOOL openHID(String devicePath)

This function is used to open the HID Device to access HID extension settings of the camera, if HID is supported by the camera.

Parameters	Description	Return Values
`String devicePath`	Camera Device path should be passed as the parameter to open the HID device access.	TRUE on Success (HID supported Camera) FALSE on Failure (HID is not supported)

Sample Code

```
VideoCapture cap;
cap.open(0);
videoDevicePath = "\dev\hidrawX";
if(cap.openHID(videoDevicePath))
{
      cout << "HID Device Access is granted" << endl;
}
```

BOOL writeData(std::vector <unsigned char> &inBuffer)

This function is used to write data to the HID device. Before calling this function, openHID() API must be called to access HID related settings. writeData is used to send a request about the HID control values.

0x40 - To read Firmware Revision Number.

Parameters	Description	Return Values
`Std::vector <unsigned char> &inBuffer`	writeData API is used to send request to the HID Device about the HID control settings.	TRUE on Success FALSE on Failure

Sample Code

```
VideoCapture cap;
cap.open(0);
if(cap.openHID(videoDevicePath))
```

```
{
      inBuffer.push_back(0x00);
      inBuffer.push_back(0x40);
      if(cap.writeData(inBuffer))
      {
            cout << "Data is written into the hid device "
<< endl;
      }
}
```

BOOL readData(std::vector <unsigned char> &outBuffer);

This function is used to read data from the HID device. Before calling this API, the camera must be opened and writeData API must be called. Using readData API, the response for the writeData will be retrieved.

0x40 - To read Firmware Revision Number.

Parameters	Description	Return Values
`Std::vector <unsigned char> &outBuffer`	Read the response from the HID device, which you must request about the HID Control settings using writeData API.	TRUE on Success FALSE on Failure

Windows Sample Code

```
VideoCapture cap;
cap.open(0);
if(cap.openHID(videoDevicePath))
{
      inBuffer.push_back(0x00);
      inBuffer.push_back(0x40);
      if(cap.writeData(inBuffer))
      {
            cout << "Data is written into the hid device "
<< endl;
      }
      if(cap.readData(outBuffer))
      {
            int pMajorVersion, pMinorVersion1,
pMinorVersion2, pMinorVersion3, SDK_VER, SVN_VER;
            SDK_VER = (outBuffer[4] << 8) + outBuffer[5];
            SVN_VER = (outBuffer[6] << 8) + outBuffer[7];
```

```
            pMajorVersion = outBuffer[2];
            pMinorVersion1 = outBuffer[3];
            pMinorVersion2 = SDK_VER;
            pMinorVersion3 = SVN_VER;
      }
}
```

Linux Sample Code

```
VideoCapture cap;
cap.open(0);
if(cap.openHID(videoDevicePath))
{
      inBuffer.push_back(0x00);
      inBuffer.push_back(0x40);
      if(cap.writeData(inBuffer))
      {
            cout << "Data is written into the hid device "
<< endl;
      }
      if(cap.readData(outBuffer))
      {
            int pMajorVersion, pMinorVersion1,
pMinorVersion2, pMinorVersion3, SDK_VER, SVN_VER;
            SDK_VER = (outBuffer[3] << 8) + outBuffer[4];
            SVN_VER = (outBuffer[5] << 8) + outBuffer[6];
            pMajorVersion = outBuffer[1];
            pMinorVersion1 = outBuffer[2];
            pMinorVersion2 = SDK_VER;
            pMinorVersion3 = SVN_VER;
      }
}
```

BOOL closeHID()

This function is used to close HID device access.

Parameters	Description	Return Values
None	N/A	TRUE on Success FALSE on Failure

Sample Code

```
VideoCapture cap;
cap.open(0);
if(cap.openHID(videoDevicePath))
{
      if(closeHID())
      {
            cout << "HID Device Access is closed" << endl;
      }
}
```

Support

Contact Us

If you need any support on OpenCV sample application, please contact us using the Live Chat option available on our website - https://www.e-consystems.com/

Creating a Ticket

If you need to create a ticket for any type of issue, please visit the ticketing page on our website - https://www.e-consystems.com/create-ticket.asp

RMA

To know about our Return Material Authorization (RMA) policy, please visit the RMA Policy page on our website - https://www.e-consystems.com/RMA-Policy.asp

General Product Warranty Terms

To know about our General Product Warranty Terms, please visit the General Warranty Terms page on our website - https://www.e-consystems.com/warranty.asp

Revision History

Rev	Date	Description	Author
1.0	10-April-2018	Initial Draft	Chandra Sekar V