Frequently Asked Questions

Transferring image data to OpenCV

(Modifies Stream_FreerunLockBuffer sample code)

#5400-0444　Version 1.0.0.1

TOSHIBA TELI CORPORATION

Document in Japanese

このドキュメントは TeliCamSDK を使用して取得した画像データを OpenCV の画像バッファに設定する方法について記述したドキュメントです。日本語ドキュメントは英語ドキュメントの後にあります。

End of document in Japanese

This document describes how to set image data acquired using TeliCamSDK to image buffer of OpenCV library..

1. Introduction

OpenCV is a computer vision library on open source basis.

OpenCV provides “IplImage” structure for handling image data in C language, and “cv::Mat” class for handling image data in C++ language

OpenCV has “videoio” module for capturing images from cameras. But “videoio” module does support USB3 Vision and GigE Vision camera.

This document shows how to set image data, captured from USB3 Vision or GigE Vision camera using API of TeliCamSDK on Microsoft Windows, to “IplImage” structure or “cv::Mat” class object.

2. Required software

This document assumes that you use Visual studio 2015 as software development tool.
The following libraries should be installed in the PC that Visual studio 2015 is installed.

Library name	Remarks
TeliCamSDK PkgVer 2.1.1.1 or later	TeliCamSDK is downloadable from the following URL. https://www.toshiba-teli.co.jp/cgi/ss/en/service.cgi
OpenCV	Explanation of this document assumes that OpenCV version 3.4.3 is installed in the folder directly under the C drive. Building binary of OpenCV library might be required depending on combination of OpenCV version and Visual Studio version. Please refer to http://opencv.org/ about OpenCV

3. Constructing software using OpenCV library

This chapter introduces the procedure to construct software for displaying the image, acquired from a camera, in OpenCV image display window, based on Stream_FreerunLockBuffer sample code attached to TeliCamSDK PkgVer3.0.0.1 or later.

The original Stream_FreerunLockBuffer is a console application which captures images from a camera in free-run mode and shows value of top 8 pixels in the received images.

The modified Stream_FreerunLockBuffer will convert captured image from the camera to format of image data used in OpenCV, and show the converted image in OpenCV image window.

In this document, code part of Stream_FreerunLockBuffer.cpp is shown enclosed in a ruled line. The code stated in a pale color such as gray is a code unnecessary to change. The code written in bold and in dark color will be the code part added or modified.

3.1. Copying Stream_FreerunLockBuffer sample code

Stream_FreerunLockBuffer sample code is arranged in the following directory as Visual Studio 2005 solution.

[Directory that TeliCamSDK is installed]/Samples/VS2005/CPP/Stream_FreerunLockBuffer

- Copy Stream_FreerunLockBuffer sample code to an arbitrary folder.
  Please copy to a folder that login user can read and write files.

- Open the copied Stream_FreerunLockBuffer.sin using Visual Studio.
  Visual Studio conversion wizard dialog will appear.

- Click [OK] button in the dialog.
  Visual Studio 2005 format solution files will be converted to Visual Studio 2015 format solution files.

- Select “Debug”→”Start Without Debugging” to make sure that conversion succeeded.
  Connect a camera to the PC and select "Start Without Debugging". Command prompt window will appear after compiling the converted solution. Confirm that Stream_FreerunLockBuffer process works correctly.

  On pressing 0 key after operation guide "Press ‘0’ key to grab frames” is shown in Command prompt, the operation guide might be shown again without showing pixel values of received images.
  In this case, open Stream_FreerunLockBuffer.cpp and insert the following codes before “while(1)” row (165th row) and rebuild the project. Pixel values of received images will be shown by pressing 0 key.

```
while (_kbhit())
{
    _getch();
}
```

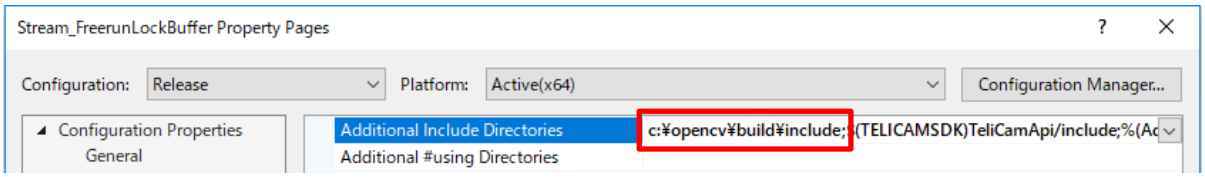

3.2. Making OpenCV library available in Stream_FreerunLockBuffer project

Modify property of Stream_FreerunLockBuffer project to make header files and Lib files of OpenCV library readable from Stream_FreerunLockBuffer project.

- Right click Stream_FreerunLockBuffer project in Solution Explorer to show property window of it.

- Add the directory which contains include file of OpenCV library to “Additional include Directories” under “Configuration Properties”→”C/C++”→”General”

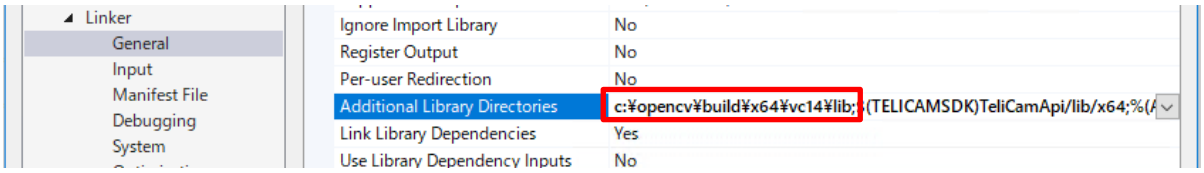
  Add “c:/opencv/build/include;” when OpenCV files are installed in “c:\opencv\”.



- Add the directory which contains library files of OpenCV to “Additional Library Directories” under “Configuration Properties”→”Linker”→”General”.

  Lib files compiled for each version of Visual Studio exist in the folder that OpenCV version 3.4.3 is installed. Add directory that corresponds to the Visual Studio you use to "Additional Library Directories".

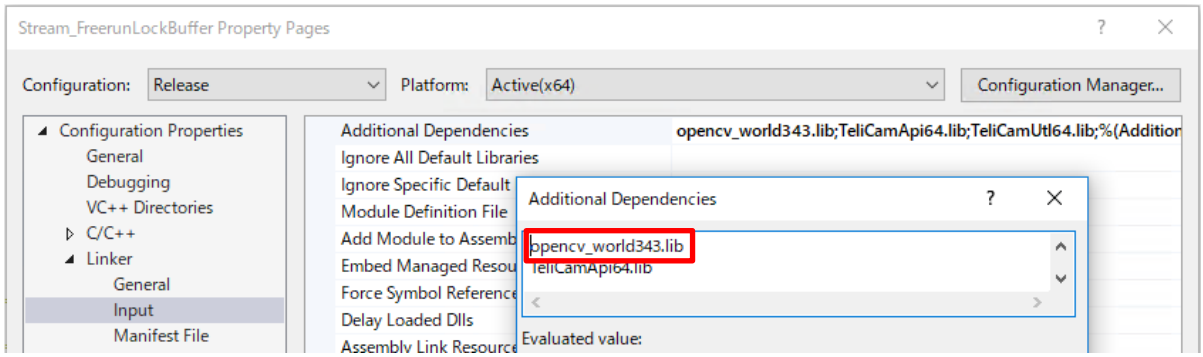
	32bit (x86)	64bit (x64)
Visual Studio 2015	c:/opencv/build/x86/vc14/lib	c:/opencv/build/x64/vc14/lib
Visual Studio 2017	c:/opencv/build/x86/vc15/lib	c:/opencv/build/x64/vc15/lib



- Add library files to “Additional Dependencies” under “Configuration Properties”→”Linker”→”Input”.

  Add the following libraries that are necessary in handling image data in OpenCV, in this document.

  opencv_world343.lib (opencv_world343d.lib in debug configuration)



3.3. Adding #Include directive for OpenCV in Stream_FreerunLockBuffer.cpp

- Open Stream_FreerunLockBuffer.cpp and add #include directive for opencv.hpp.

  Open Stream_FreerunLockBuffer.cpp and add directive for including opencv2/opencv.hpp before "#include "TeliCamApi.h" row to make classes and functions in OpenCV available.

```
#include "opencv2/opencv.hpp"
```

3.4. Adding declaration of OpenCV image data object

- Add declaration of OpenCV image data object in Stream_FreerunLockBuffer.cpp.

  In Stream_FreerunLockBuffer, you should decide which one of IplImage structure or cv::Mat class object to use. When you use IplImage structure, delete "//" from the first line "// # define USE_IPL_IMAGE" to activate the #define preprocessor directive.

```
//#define USE_IPL_IMAGE

#ifdef USE_IPL_IMAGE
static ::IplImage        *s_pIplImage;
#else
static cv::Mat           s_openCvImage;
#endif
```

3.5. Modifying `_tmain()` function

Please add codes for creating / releasing OpenCV image object and replace code for calling `ProcessLoop()` with code for calling `ImageHandling()`.

`ProcessLoop()` was a function to show operation guide and process key input.
`ImageHandling()` will be modified to show received images in OoenCV image window in the next secton.

3.5.1. Adding codes for allocating memory to OpenCV image object.

Add code for allocating memory to OpenCV image object after the row calling `Strm_OpenSimple()`.

Also add code for getting image width and image height before code for allocating memory, because function of OpenCV for allocating memory requires image width and image height information.

Code of this section sets 3 to the argument of 'number of channels' assuming that BGR format is used in the OpenCV image processing. In the actual application, value of 'number of channels' argument should be decided depending on the image data format used for image processing.

3.5.2. Calling `ImageHandling()` instead of `ProcessLoop()`

This modification changes the behavior of the application to show an OpenCV image window on pressing any key at the beginning of the application and to show received images in the image window.

ImageHandling() will be modified in section 3.6 to show OpenCV image window at the beginning of the function and to close the image window when Escape key is pressed. ImageHandling() will finished immediately after the image window is closed.

Add code for waiting for key input so that error comment can be confirmed when `ImageHandleng()` terminates abnormally.

3.5.3. Adding codes for releasing OpenCV image object

Add code for releasing OpenCV image object after the row calling `ImageHandling()`.

```
int _tmain(int argc, _TCHAR* argv[])
{
    CAM_API_STATUS  uiStatus     = CAM_API_STS_SUCCESS;
    bool8_t         bStatus      = true;
    uint32_t        uiNum        = 0;
    uint32_t        uiImgBufSize = 0;       // Size of image buffer.

    printf("Stream_FreerunLockBuffer sample application for TeliCamSDK and OpenCV.\n\n");
    printf("Copyrights (C) 2014 - 2018 TOSHIBA TELI CORPORATION. All rights reserved.\n\n");
    printf("Press any key to start...\n");
    _getch();

    do
    {
        // API initialization.
        uiStatus = Sys_Initialize();
        if (uiStatus != CAM_API_STS_SUCCESS)
        {
            printf("Failed Sys_Initialize, Status = 0x%08X.\n", uiStatus);
            break;
        }

        // Get number of camera.
        uiStatus = Sys_GetNumOfCameras(&uiNum);
        if (uiStatus != CAM_API_STS_SUCCESS)
        {
            printf("Failed Sys_GetNumOfCameras, Status = 0x%08X.\n", uiStatus);
            break;
        }
        if (uiNum == 0)
        {
            printf(" No cameras found.\n");
            break;
        }

        // Open camera that is detected first, in this sample code.
        uiStatus = Cam_Open(0, &s_hCam);
        if (uiStatus != CAM_API_STS_SUCCESS)
        {
            printf("Failed Cam_Open, Status = 0x%08X.\n", uiStatus);
            break;
        }

        // Create completion event object for stream.
        s_hStrmCmpEvt = CreateEvent(NULL, FALSE, FALSE, NULL);
        if (s_hStrmCmpEvt == NULL)
        {
            printf("Failed CreateEvent.\n");
            break;
        }

        // Open stream.
        uiStatus = Strm_OpenSimple(s_hCam, &s_hStrm, &uiImgBufSize, s_hStrmCmpEvt);
        if (uiStatus != CAM_API_STS_SUCCESS)
        {
            printf("Failed Strm_OpenSimple, Status = 0x%08X.\n", uiStatus);
            break;
        }

        // Allocate memory to OpenCV image object.
        //   Gets width and height of image.
        uint32_t uiWidth, uiHeight;

        uiStatus = GetCamWidth(s_hCam, &uiWidth);
        if (uiStatus != CAM_API_STS_SUCCESS)
        {
            printf("Failed GetCamWidth, Status = 0x%08X.\n", uiStatus);
            break;
        }
```

3.5.1

```
        uiStatus = GetCamHeight(s_hCam, &uiHeight);
        if (uiStatus != CAM_API_STS_SUCCESS)
        {
            printf("Failed GetCamHeight, Status = 0x%08X.\n", uiStatus);
            break;
        }

        //    Allocate memory to OpenCV image object.
#ifdef USE_IPL_IMAGE
        s_pIplImage = ::cvCreateImage(::cvSize(uiWidth, uiHeight), IPL_DEPTH_8U, 3);
#else
        s_openCvImage = cv::Mat(cv::Size(uiWidth, uiHeight), CV_8UC3, cv::Scalar::all(0));
#endif

       // Set free-run mode.
       // Set TriggerMode false.
       uiStatus = SetCamTriggerMode(s_hCam, false);
       if (uiStatus != CAM_API_STS_SUCCESS)
       {
           printf("Failed SetCamTriggerMode, Status = 0x%08X.\n", uiStatus);
           break;
       }

       // Receive images.
        // bStatus = ProcessLoop();
       //if (!bStatus)
       //{
       //    break;
       //}

       // Receive images.
        if (!ImageHandling())
        {
            printf("Press any key to exit...\n");
            _getch();
        }

    } while (0);

    // Close stream.
    if (s_hStrm != (CAM_STRM_HANDLE)NULL)
    {
        Strm_Close(s_hStrm);
        s_hStrm = (CAM_STRM_HANDLE)NULL;
    }

    if (s_hStrmCmpEvt != NULL)
    {
        CloseHandle(s_hStrmCmpEvt);
        s_hStrmCmpEvt = NULL;
    }

    // Release image buffer of OpenCV image object.
#ifdef USE_IPL_IMAGE
    if (s_pIplImage != NULL)
    {
        ::cvReleaseImage(&s_pIplImage);
        s_pIplImage = NULL;
    }
#else
    // Do Nothing
#endif

    // Close camera.
    if (s_hCam != (CAM_HANDLE)NULL)
    {
        Cam_Close(s_hCam);
        s_hCam = (CAM_HANDLE)NULL;
    }
```

3.5.1

3.5.2

3.5.3


```
    // Terminate system.
    Sys_Terminate();
    //printf("Finished.\n\n");
    //printf("Press any key to exit...\n");
    //_getch();

    return 0;
}
```

3.6. Modifying `ImageHandling()` function

Please add codes for creating / releasing OpenCV image window, code for show received images in the image window and code for monitoring key input to close the image window.

3.6.1. Adding code for creating OpenCV image window

Add code for creating OpenCV image window at the beginning of `ImageHandling()` function.
Also, add declaration of a variable 'key' used in monitoring key input procedure here.

3.6.2. Commenting out code for checking key input at the beginning of while loop

Codes for monitoring key input at the beginning of the while loop is not used in the modified `ImageHandling()`. Please comment them out.

3.6.3. Adding code for writing BGR format image data to OpenCV image object.

Add code for writing BGR format image data converted from an image data received from the camera using camera utility function `ConvImage()` to OpenCV image object. The member variable values of image information structure 'imageInfo' got by `Strm_LockBuffer()` are used for arguments of ConvImage() as source image data, width, height and PixelFormat of the image data.

This example code specifies PixelFormat member variable value in the image information structure to the 2nd argument of `ConvImage()`.In some firmware version of some camera model, this example code might cause error in ConvImage() function or might convert the image improperly.

In such a case, please specify the PixelFormat value obtained by the `GetCamPixelFormat ()` function instead of the PixelFormat member variable of image information

3.6.4. Adding code for showing the image in OpenCV image window.

Add code to show image in OpenCV image window using `::cvShowImage()` or `cv::imshow()`.
The specified image data will be drawn when `waitkey()` function is executed.

3.6.5. Add code for breaking "while" loop.

Add code for breaking "while" loop when Escape key is pressed.

3.6.6. Comment out code for printing pixel value of the received image

The modified `ImageHandling()` does not print pixel value in the command prompt.

3.6.7. Add code for releasing OpenCV image window

Add code for releasing OpenCV image window at the end of `ImageHandling()` function.

```
bool8_t ImageHandling()
{
    CAM_API_STATUS  uiStatus      = CAM_API_STS_SUCCESS;
    uint32_t        uiRes;
    uint32_t        uiBufferIndex = 0;
    uint8_t        *pImageBuf     = NULL;
    bool8_t         bStatus       = true;
    CAM_IMAGE_INFO  imageInfo;

    // Declare key value variable for getting pressed key.
    int             key;
    // Create OpenCV window for showing image.
    cv::namedWindow("OpenCV Test", CV_WINDOW_AUTOSIZE);

    // Start stream.
    uiStatus = Strm_Start(s_hStrm);
    if (uiStatus != CAM_API_STS_SUCCESS)
    {
         bStatus = false;
         printf("Failed Strm_Start, Status = 0x%08X.\n", uiStatus);
        return false;
    }

    while (_kbhit())
    {
         _getch();
    }

    while (1)
    {
         //if (_kbhit())
         //{
         //    _getch();
         //    break;
         //}

         // Wait for receiving image completion event.
         uiRes = WaitForSingleObject(s_hStrmCmpEvt, 2000);
         if (uiRes != WAIT_OBJECT_0)
         {
              bStatus = false;
              printf("Timeout WaitForSingleObject.\n");
              break;
         }

         // Get current ring buffer index.
         uiStatus = Strm_GetCurrentBufferIndex(s_hStrm, &uiBufferIndex);
         if (uiStatus != CAM_API_STS_SUCCESS)
         {
              bStatus = false;
              printf("Failed Strm_GetCurrentBufferIndex, Status = 0x%08X.\n", uiStatus);
              break;
         }

         // Lock the ring buffer pointer.
         uiStatus = Strm_LockBuffer(s_hStrm, uiBufferIndex, &imageInfo);
         if (uiStatus != CAM_API_STS_SUCCESS)
         {
              bStatus = false;
              printf("Failed Strm_LockBuffer, Status = 0x%08X.\n", uiStatus);
              break;
         }

#ifdef USE_IPL_IMAGE
         if (PXL_FMT_Mono8 == s_pixelFormat)
         {
              // Copy received image data to OpenCV image object.
              rsize_t size = imageInfo.uiSizeX * imageInfo.uiSizeY;
              memcpy_s(s_pIplImage->imageData, size, imageInfo.pvBuf, size);
         }
```

3.6.1.

3.6.2.

3.6.3.


```
        else
        {
            // Copy converted image data to OpenCV image object.
            uiStatus = ConvImage(DST_FMT_BGR24, imageInfo.uiPixelFormat, true,
                s_pIplImage->imageData, imageInfo.pvBuf,
                imageInfo.uiSizeX, imageInfo.uiSizeY);
            if (uiStatus != CAM_API_STS_SUCCESS)
            {
                //bStatus = false;
                printf("Failed ConvImage, Status = 0x%08X.\n", uiStatus);
                break;
            }
        }

        // Show image in OpenCV window.
        ::cvShowImage("OpenCV Test", s_pIplImage);
#else
        if (PXL_FMT_Mono8 == s_pixelFormat)
        {
            // Copy received image data to OpenCV image object.
            rsize_t size = imageInfo.uiSizeX * imageInfo.uiSizeY;
            memcpy_s(s_openCvImage.data, size, imageInfo.pvBuf, size);
        }
        else
        {
            // Copy converted image data to OpenCV image object.
            uiStatus = ConvImage(DST_FMT_BGR24, imageInfo.uiPixelFormat, true,
                                 s_openCvImage.data, imageInfo.pvBuf,
                                 imageInfo.uiSizeX, imageInfo.uiSizeY);
            if (uiStatus != CAM_API_STS_SUCCESS)
            {
                bStatus = false;
                printf("Failed ConvImage, Status = 0x%08X.\n", uiStatus);
                break;
            }
        }

        // Show image in OpenCV window.
        cv::imshow("OpenCV Test", s_openCvImage);
#endif

        // Add code for checking that Escape key is pressed.
        key = cv::waitKey(10);
        if (key == 0x1b)    // [ESC] key
        {
            break;
        }

       //// Display the first 8 pixel values(8 bit color).
       //pImageBuf = (uint8_t*)imageInfo.pvBuf;
       //printf("  Image :");
       //for (int i = 0; i < 8; i++)
       //{
       //    printf(" %02X ", pImageBuf[i]);
       //}
       //printf("...\n");

        // Unlock the ring buffer pointer.
        uiStatus = Strm_UnlockBuffer(s_hStrm, uiBufferIndex);
        if (uiStatus != CAM_API_STS_SUCCESS)
        {
            bStatus = false;
            printf("Failed Strm_UnlockBuffer, Status = 0x%08X.\n", uiStatus);
            break;
        }
    }

    // Destroy OpenCV image window.
    cv::destroyWindow("OpenCV Test");
```

3.6.3.
3.6.4.
3.6.3
3.6.4.
3.6.5.
3.6.6.
3.6.7.

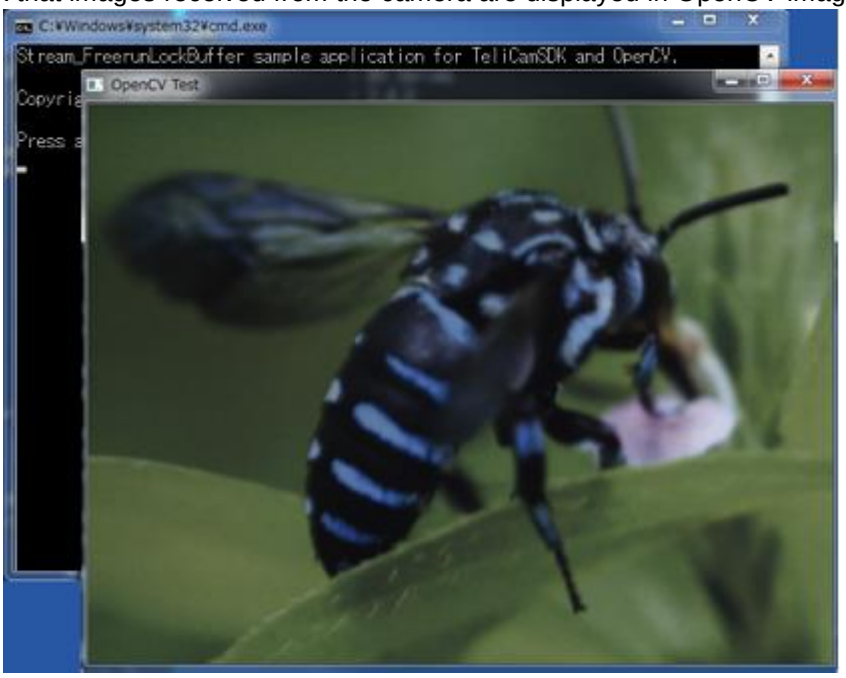
```
    //Stop stream.
    uiStatus = Strm_Stop(s_hStrm);
    if (uiStatus != CAM_API_STS_SUCCESS)
    {
        bStatus = false;
        printf("Failed Strm_Stop, Status = 0x%08X.\n", uiStatus);
    }

    Sleep(100);

    return bStatus;
}
```

3.7. Deleting ProcessLoop() function

The modified Stream_FreerunLockBuffer does not use the original `ProcessLoop ()` function. Please delete it.

3.8. Checking that Stream_FreerunLockBuffer is modified correctly

After finished editing Stream_FreerunLockBuffer project, select “Debug”->”Start Without Debugging” to check that images received from the camera are displayed in OpenCV image window.



If error message of “no OpenCV DLL found”is displayed, path of OpenCV DLL file might not be set in the environment variable PATH.

Please add folder that contains path OpenCV DLLs for Visual Studio of the version you use to the environment variable PATH.

	32bit (x86)	64bit (x64)
Visual Studio 2015	c:/opencv/build/x86/vc14/bin	c:/opencv/build/x64/vc14/bin
Visual Studio 2017	c:/opencv/build/x86/vc15/bin	c:/opencv/build/x64/vc15/bin

The followings show procedure for editing the environment variable PATH.

- ✓ Run “Control Panel” →”System”→”Advanced system settings” to open “System Properties” window.
- ✓ Click [Environment Variables] button to show “Environment Variables” window.
- ✓ Scroll down “System variables” table, select variable “Path” and click [Edit] button to open “Edit System variable” window.
- ✓ Add folder name to value shown in “Variable value:” with semicolon as a separator.
- ✓ Close “Edit System variable” window and other opened windows by clicking [OK] button.

3.9. Stream_FreerunLockBuffer.cpp after modification

This section shows the modified Stream_FreerunLockBuffer.cpp. Commented out codes are not shown in this section.

The following modifications are applied in addition to the modification described in the section above.

- Specify 1 to channel count of OpenCV image object when PixelFormat sent from the camera is Mono8.

When PixelFormat sent from the camera is Mono8, copies the received image data to OpenCV image object as is.

```
#include <windows.h>
#include <tchar.h>
#include <conio.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#include "opencv2/opencv.hpp"

#include "TeliCamApi.h"
#include "TeliCamUtl.h"

using namespace Teli;

//=============================================================================
// Prototype declares                                                         
//=============================================================================

bool8_t ImageHandling();

//=============================================================================
// Static Values                                                              
//=============================================================================

// Handles
static CAM_HANDLE       s_hCam        = (CAM_HANDLE)NULL;              // Camera handle.
static CAM_STRM_HANDLE  s_hStrm       = (CAM_STRM_HANDLE)NULL;         // Stream handle.
static HANDLE           s_hStrmCmpEvt = NULL;                          //event object for stream.
static CAM_PIXEL_FORMAT s_pixelFormat;

//#define USE_IPL_IMAGE
#ifdef USE_IPL_IMAGE
static ::IplImage        *s_pIplImage;
#else
static cv::Mat           s_openCvImage;
#endif

//=============================================================================
// Functions                                                                  
//=============================================================================
int _tmain(int argc, _TCHAR* argv[])
{
    CAM_API_STATUS  uiStatus     = CAM_API_STS_SUCCESS;
    bool8_t         bStatus      = true;
    uint32_t        uiNum        = 0;
    uint32_t        uiImgBufSize = 0;       // Size of image buffer.

    printf("Stream_FreerunLockBuffer sample application for TeliCamSDK and OpenCV.\n\n");
    printf("Copyrights (C) 2014 - 2018 TOSHIBA TELI CORPORATION. All rights reserved.\n\n");
    printf("Press any key to start...\n");
    _getch();

    do
    {
```

```
        // API initialization.
        uiStatus = Sys_Initialize();
        if (uiStatus != CAM_API_STS_SUCCESS)
        {
            printf("Failed Sys_Initialize, Status = 0x%08X.\n", uiStatus);
            break;
        }

        // Get number of camera.
        uiStatus = Sys_GetNumOfCameras(&uiNum);
        if (uiStatus != CAM_API_STS_SUCCESS)
        {
            printf("Failed Sys_GetNumOfCameras, Status = 0x%08X.\n", uiStatus);
            break;
        }
        if (uiNum == 0)
        {
            printf(" No cameras found.\n");
            break;
        }

        // Open camera that is detected first, in this sample code.
        uiStatus = Cam_Open(0, &s_hCam);
        if (uiStatus != CAM_API_STS_SUCCESS)
        {
            printf("Failed Cam_Open, Status = 0x%08X.\n", uiStatus);
            break;
        }

        // Create completion event object for stream.
        s_hStrmCmpEvt = CreateEvent(NULL, FALSE, FALSE, NULL);
        if (s_hStrmCmpEvt == NULL)
        {
            printf("Failed CreateEvent.\n");
            break;
        }

        // Open stream.
        uiStatus = Strm_OpenSimple(s_hCam, &s_hStrm, &uiImgBufSize, s_hStrmCmpEvt);
        if (uiStatus != CAM_API_STS_SUCCESS)
        {
            printf("Failed Strm_OpenSimple, Status = 0x%08X.\n", uiStatus);
            break;
        }

         // Allocate memory to OpenCV image object.
         //   Gets width, height and PixelFormat of image.
         uint32_t uiWidth, uiHeight;

         uiStatus = GetCamWidth(s_hCam, &uiWidth);
         if (uiStatus != CAM_API_STS_SUCCESS)
         {
             printf("Failed GetCamWidth, Status = 0x%08X.\n", uiStatus);
             break;
         }

         uiStatus = GetCamHeight(s_hCam, &uiHeight);
         if (uiStatus != CAM_API_STS_SUCCESS)
         {
             printf("Failed GetCamHeight, Status = 0x%08X.\n", uiStatus);
             break;
         }

         uiStatus = GetCamPixelFormat(s_hCam, &s_pixelFormat);
         if (uiStatus != CAM_API_STS_SUCCESS)
         {
             printf("Failed GetCamPixelFormat, Status = 0x%08X.\n", uiStatus);
             break;
         }
```

```
        //    Allocate memory to OpenCV image object.
#ifdef USE_IPL_IMAGE
        if (PXL_FMT_Mono8 == s_pixelFormat)
        {
            s_pIplImage = ::cvCreateImage(::cvSize(uiWidth, uiHeight), IPL_DEPTH_8U, 1);
        }
        else
        {
            s_pIplImage = ::cvCreateImage(::cvSize(uiWidth, uiHeight), IPL_DEPTH_8U, 3);
        }

#else
        if (PXL_FMT_Mono8 == s_pixelFormat)
        {
            s_openCvImage = cv::Mat(cv::Size(uiWidth, uiHeight), CV_8UC1, cv::Scalar::all(0));
        }
        else
        {
            s_openCvImage = cv::Mat(cv::Size(uiWidth, uiHeight), CV_8UC3, cv::Scalar::all(0));
        }
#endif
       // Set free-run mode.
       // Set TriggerMode false.
       uiStatus = SetCamTriggerMode(s_hCam, false);
       if (uiStatus != CAM_API_STS_SUCCESS)
       {
           printf("Failed SetCamTriggerMode, Status = 0x%08X.\n", uiStatus);
           break;
       }

       // Receive images.
        if (!ImageHandling())
        {
            printf("Press any key to exit...\n");
            _getch();
        }
    } while (0);

    // Close stream.
    if (s_hStrm != (CAM_STRM_HANDLE)NULL)
    {
        Strm_Close(s_hStrm);
        s_hStrm = (CAM_STRM_HANDLE)NULL;
    }

    if (s_hStrmCmpEvt != NULL)
    {
        CloseHandle(s_hStrmCmpEvt);
        s_hStrmCmpEvt = NULL;
    }

    // Release image buffer of OpenCV image object.
#ifdef USE_IPL_IMAGE
    if (s_pIplImage != NULL)
    {
        ::cvReleaseImage(&s_pIplImage);
        s_pIplImage = NULL;
    }
#else
    // Do Nothing
#endif

    // Close camera.
    if (s_hCam != (CAM_HANDLE)NULL)
    {
        Cam_Close(s_hCam);
        s_hCam = (CAM_HANDLE)NULL;
    }

    // Terminate system.
```



```
    Sys_Terminate();

    return 0;
}

////////////////////////////////////////////////////////
//
bool8_t ImageHandling()
{
    CAM_API_STATUS  uiStatus     = CAM_API_STS_SUCCESS;
    uint32_t        uiRes;
    uint32_t        uiBufferIndex = 0;
    uint8_t        *pImageBuf    = NULL;
    bool8_t         bStatus      = true;
    CAM_IMAGE_INFO  imageInfo;

    // Declare key value variable for getting pressed key.
    int             key;
    // Create OpenCV window for showing image.
    cv::namedWindow("OpenCV Test", CV_WINDOW_AUTOSIZE);

    // Start stream.
    uiStatus = Strm_Start(s_hStrm);
    if (uiStatus != CAM_API_STS_SUCCESS)
    {
        bStatus = false;
        printf("Failed Strm_Start, Status = 0x%08X.\n", uiStatus);
       return false;
    }

    while (_kbhit())
    {
        _getch();
    }

    while (1)
    {
        // Wait for receiving image completion event.
        uiRes = WaitForSingleObject(s_hStrmCmpEvt, 2000);
        if (uiRes != WAIT_OBJECT_0)
        {
            bStatus = false;
            printf("Timeout WaitForSingleObject.\n");
            break;
        }

        // Get current ring buffer index.
        uiStatus = Strm_GetCurrentBufferIndex(s_hStrm, &uiBufferIndex);
        if (uiStatus != CAM_API_STS_SUCCESS)
        {
            bStatus = false;
            printf("Failed Strm_GetCurrentBufferIndex, Status = 0x%08X.\n", uiStatus);
            break;
        }

        // Lock the ring buffer pointer.
        uiStatus = Strm_LockBuffer(s_hStrm, uiBufferIndex, &imageInfo);
        if (uiStatus != CAM_API_STS_SUCCESS)
        {
            bStatus = false;
            printf("Failed Strm_LockBuffer, Status = 0x%08X.\n", uiStatus);
            break;
        }

#ifdef USE_IPL_IMAGE
        if (PXL_FMT_Mono8 == s_pixelFormat)
        {
            // Copy received image data to OpenCV image object.
            rsize_t size = imageInfo.uiSizeX * imageInfo.uiSizeY;
            memcpy_s(s_pIplImage->imageData, size, imageInfo.pvBuf, size);
```



```
        }
        else
        {
            // Copy converted image data to OpenCV image object.
            uiStatus = ConvImage(DST_FMT_BGR24, imageInfo.uiPixelFormat, true,
                s_pIplImage->imageData, imageInfo.pvBuf,
                imageInfo.uiSizeX, imageInfo.uiSizeY);
            if (uiStatus != CAM_API_STS_SUCCESS)
            {
                //bStatus = false;
                printf("Failed ConvImage, Status = 0x%08X.\n", uiStatus);
                break;
            }
        }

        // Show image in OpenCV window.
        ::cvShowImage("OpenCV Test", s_pIplImage);
#else
        if (PXL_FMT_Mono8 == s_pixelFormat)
        {
            // Copy received image data to OpenCV image object.
            rsize_t size = imageInfo.uiSizeX * imageInfo.uiSizeY;
            memcpy_s(s_openCvImage.data, size, imageInfo.pvBuf, size);
        }
        else
        {
            // Copy converted image data to OpenCV image object.
            uiStatus = ConvImage(DST_FMT_BGR24, imageInfo.uiPixelFormat, true,
                                 s_openCvImage.data, imageInfo.pvBuf,
                                 imageInfo.uiSizeX, imageInfo.uiSizeY);
            if (uiStatus != CAM_API_STS_SUCCESS)
            {
                bStatus = false;
                printf("Failed ConvImage, Status = 0x%08X.\n", uiStatus);
                break;
            }
        }
        // Show image in OpenCV window.
        cv::imshow("OpenCV Test", s_openCvImage);
#endif

        // Add code for checking that Escape key is pressed.
        key = cv::waitKey(10);
        if (key == 0x1b)    // [ESC] key
        {
            break;
        }

       // Unlock the ring buffer pointer.
       uiStatus = Strm_UnlockBuffer(s_hStrm, uiBufferIndex);
       if (uiStatus != CAM_API_STS_SUCCESS)
       {
           bStatus = false;
           printf("Failed Strm_UnlockBuffer, Status = 0x%08X.\n", uiStatus);
           break;
       }
    }

    // Destroy OpenCV image window.
    cv::destroyWindow("OpenCV Test");

    //Stop stream.
    uiStatus = Strm_Stop(s_hStrm);
    if (uiStatus != CAM_API_STS_SUCCESS)
    {
        bStatus = false;
        printf("Failed Strm_Stop, Status = 0x%08X.\n", uiStatus);
    }

    Sleep(100);
```



```
    return bStatus;
}
```


Appendix 1. Using Other version OpenCV

If you use OpenCV 2.x or OpenCV 3.x that provides binary files for Visual studio, please edit "Additional Library Directories" and "Additional Dependencies" as described in this chapter and edit the other items as described in Chapter 3. The modified Stream_FreerunLockBuffer will succeed in showing received images in an OpenCV image window.

Appendix 1.1. Editing “Additional Library Directories”

Version of Visual studio that OpenCV binary files (*.lib) are provided depends on the version of OpenCV. Please add the folder path that OpenCV binary files (*.lib) are stored.

	32bit (x86)	64bit (x64)
Visual Studio 2010	c:/opencv/build/x86/vc10/lib	c:/opencv/build/x64/vc10/lib
Visual Studio 2012	c:/opencv/build/x86/vc11/lib	c:/opencv/build/x64/vc11/lib
Visual Studio 2013	c:/opencv/build/x86/vc12/lib	c:/opencv/build/x64/vc12/lib
Visual Studio 2015	c:/opencv/build/x86/vc14/lib	c:/opencv/build/x64/vc14/lib
Visual Studio 2017	c:/opencv/build/x86/vc15/lib	c:/opencv/build/x64/vc15/lib

If OpenCV binary files for the version of Visual Studio that you use is not provided, you can use OpenCV binary files for the version other than the version of Visual Studio that you use.

Note that if you build an application using OpenCV binary files compiled with the Visual Studio version other than that of the Visual Studio used for compiling, VC++ runtime for the version of Visual Studio that the OpenCV binary files was compiled is necessary for executing the compiled application.

Appendix 1.2. Editing “Additional Dependencies”

Appendix 1.2.1 OpenCV3.x

Add OpenCV library file name to “Additional Dependencies”.

The library file name is the name obtained by replacing "**xxx**" in "opencv_world**xxx**.lib" with a string concatenated with three numbers constituting the version of OpenCV.

Appendix 1.2.2 OpenCV2.x

Add three library file names to “Additional Dependencies”. The file names are obtained by replacing “**xxx**” in the following file names with a string concatenated with three numbers constituting the version of OpenCV.

opencv_core**xxx**.lib
opencv_highgui**xxx**.lib
opencv_imgproc**xxx**.lib

1. はじめに

OpenCV は、オープンソースのコンピュータ・ビジョンライブラリです。

OpenCV は C 言語で画像データを扱うための IplImage 構造体と、C++言語で画像データを扱うための cv::Mat クラスを提供しています。

OpenCV にはカメラから画像を取得することができる videoio モジュールを持っていますが、このモジュールは USB3 Vision と GigE Vision の カメラ規格をサポートしておりません。

本ドキュメントは、Microsoft Windows で TeliCamSDK を使用し、USB3 Vision カメラおよび GigE Vision カメラの画像データを OpenCV の IplImage 構造体または cv::Mat クラスオブジェクトとして取り込む方法について説明します。

2. 使用ソフトウェア

本ドキュメントはソフトウェア開発ツールとして Visual Studio 2015 を使用する前提で説明を記載しています。

Visual Studio 2015 がインストールされている PC には以下のライブラリがインストールされている必要があります.

ライブラリ名	備　　考
TeliCamSDK PkgVer 2.2.1.1 or later	TeliCamSDK は以下の URL からダウンロードできます。 https://www.toshiba-teli.co.jp/cgi/ss/jp/service_j.cgi
OpenCV	この文書では OpenCV version 3.4.3 が C ドライブの直下のフォルダにインストールされている前提で説明をしています。 ご使用の OpenCV のバージョンと Visual Studio のバージョンの組み合わせによっては OpenCV ライブラリのビルドが必要になる場合があります。 OpenCV についての情報は http://opencv.org/ をご覧ください。

3. OpenCV ライブラリを使用した画像取得ソフトの作成

この章では、カメラから取得した画像を OpenCV の画像表示ウィンドウに表示するソフトウェアを、PkgVer3.0.0.1 以降の TeliCamSDK をインストールするとインストールされる Stream_FreeRunLockBuffer サンプルコードをもとにして作成する手順を紹介します。

Stream_FreeRunLockBuffer サンプルコードは、フリーラン動作でカメラから取得した画像の先頭８倍との値を画面に表示するコンソールアプリケーションです。

このアプリケーションを、カメラから取得した画像データを OpenCV の IplImage 構造体または cv::Mat クラスオブジェクトのフォーマットに変換し、OpenCV の画像表示ウィンドウに表示するよう改造します。

このドキュメントでは 3.3 節以降に Stream_FreeRunLockBuffer.cpp のコードを抜粋し罫線で囲って掲載しています。灰色などの淡い色で記載されたコードは変更不要なコードです。濃い色の太字で記載されたコードが追加または変更するコード部分になります。

3.1. Stream_FreerunLockBuffer サンプルコードのコピーとオープン

以下のディレクトリに Visual Studio 2005 のソリューションの形で Stream_FreeRunLockBuffer サンプルコードが配置されています。

[TeliCamSDK インストールディレクトリ]\Samples\VS2005\CPP\Stream_FreeRunLockBuffer

- Stream_FreerunLockBuffer サンプルコードを別のフォルダにコピーしてください。
  読み書き可能なフォルダにコピーしてください。

- Visual Studio で、コピーした Stream_FreeRunLockBuffer フォルダ内の StreamFreeRunLockBuffer_.sln を開いてください。
  「プロジェクトとソリューションの変換をレビュー」のダイアログが表示されます。

- 「プロジェクトとソリューションの変換をレビュー」ダイアログの[OK] ボタンをクリックしてください。
  Visual Studio 2005 のソリューションが Visual Studio 2015 のソリューションに変換されます。

- 「デバッグ」→「デバッグなしで実行」を選択し、正常に変換されたことを確認してください。
  カメラを PC に接続してから「デバッグなしで実行」を選択してください。ソリューションがコンパイルされた後、コンソール画面が開き、Stream_LockBuffer の処理が正常に動作すれば、変換成功です。

  ProcessLoop()開始後に 0 キーを押すと、画像データ先頭部を何も表示せずに「Press '0' key to grab frames."の操作ガイド文字が表示される場合は、Stream_FreerunLockBuffer.cpp を開き、165 行目の「while (1)」の行の前に以下のコードを挿入すれば正常動作します。

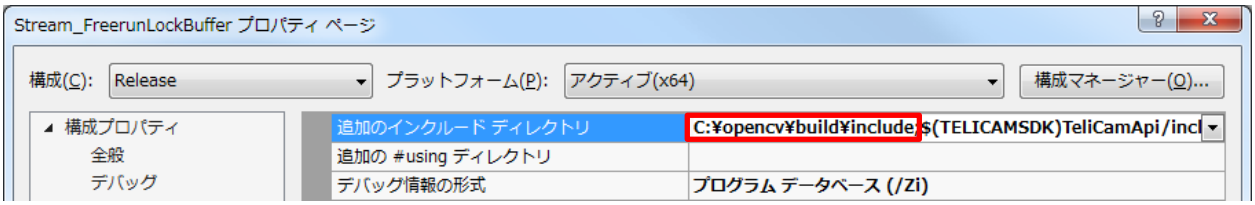
```
while (_kbhit())
{
    _getch();
}
```


3.2. Stream_FreerunLockBuffer プロジェクトで OpenCV を利用可能に設定

Stream_FreerunLockBuffer プロジェクトのプロパティで OpenCV ライブラリのヘッダファイルと Lib ファイルを読み込めるように設定します。

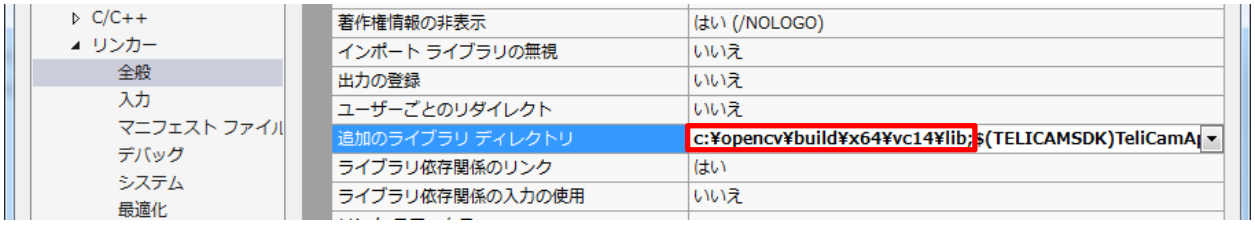
- ソリューションエクスプローラーで Stream_FreerunLockBuffer プロジェクトを右クリックし、プロジェクトのプロパティ画面を表示させてください。

- 「構成プロパティ」→「C/C++」→「 全般」の「追加のインクルード　ディレクトリ」に OpenCV ライブラリの include ファイルが格納されているディレクトリを追加してください。
  OpenCV が c:\opencv\にインストールされているときは「c:\opencv\build\include;」を追加して下さい。

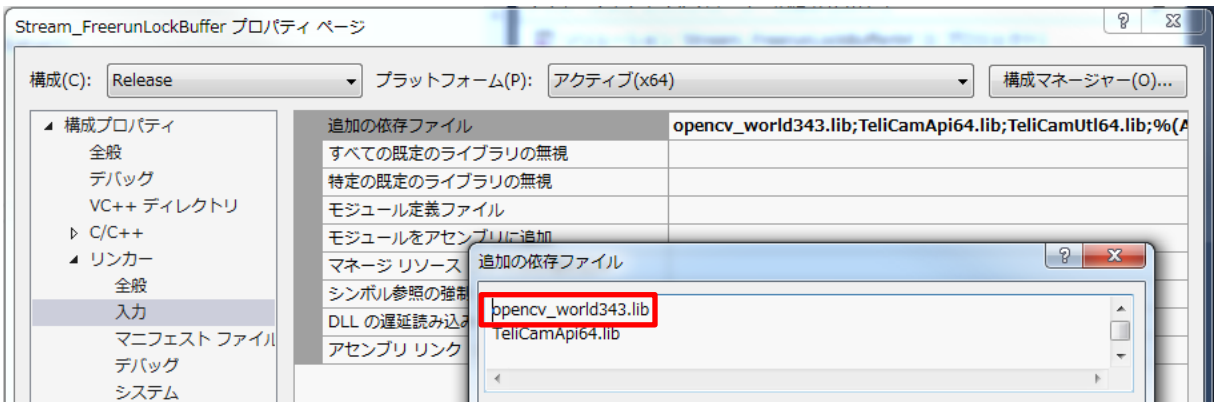


- 「構成プロパティ」→「リンカー」→「 全般」の「追加のライブラリディレクトリ」に、OpenCV のライブラリファイルが格納されているディレクトリを追加してください。
  OpenCV version 3.4.3 では Visual Studio の各バージョンにあわせてコンパイルした lib ファイルが予め提供されています。使用する Visual Studio のバージョンにあったディレクトリを追加してください。

	32bit (x86)	64bit (x64)
Visual Studio 2015	c:\opencv\build\x86\vc14\lib	c:\opencv\build\x64\vc14\lib
Visual Studio 2017	c:\opencv\build\x86\vc15\lib	c:\opencv\build\x64\vc15\lib



- 「構成プロパティ」→「リンカー」→「入力」の「追加の依存ファイル」に、使用するライブラリファイルを追加してください。
  本ドキュメントでは、OpenCV で画像を扱うために必要な以下のライブラリを追加します。
  opencv_world343.lib　（Debug コンフィギュレーションの場合は opencv_world343d.lib）



3.3. Opencv ヘッダファイルの#include ディレクティブの追加

➢ Opencv.hpp への#include ディレクティブの Stream_FreerunLockBuffer.cpp への追加

Stream_FreerunLockBuffer.cpp を開き、「`#include "TeliCamApi.h"`」の行の前に opencv2/opencv.hpp を組み込むディレクティブを追加して、OpenCV のクラスと関数を使用できるようにします。

```
#include "opencv2/opencv.hpp"
```

3.4. OpenCV 画像データを格納する変数の宣言追加

本ドキュメントでは `IplImage` 構造体と cv::Mat クラスオブジェクトのどちらか一方を宣言するようにしています。`IplImage` 構造体を使用する場合は、先頭行の「`//#define USE_IPL_IMAGE`」から[「`//`」を削除して`#define` プリプロセッサディレクティブを有効にしてください。

```
//#define USE_IPL_IMAGE

#ifdef USE_IPL_IMAGE
static ::IplImage        *s_pIplImage;
#else
static cv::Mat           s_openCvImage;
#endif
```

3.5. `_tmain()`関数の変更

`_tmain()`関数に OpenCV 画像オブジェクトを生成する処理と解放する処理を追加し、画像受信動作のためのキー受付処理を行っていた `ProcessLoop()`を廃止し、直接 `ImageHandling()`を呼ぶように変更します。

3.5.1. OpenCV 画像バッファにメモリを割付けるコードの追加

`Strm_OpenSimple()`の行の後に、OpenCV の画像バッファのメモリを確保するコードを追加してください。

OpenCV では画像バッファのメモリを確保するためには画像の縦横画素数を指定する必要があるため、画像幅と画像高さを取得するコードも追加します。

実際のアプリケーションでは画像処理で使用する画像データフォーマットに合わせてチャンネル数を設定しますが、ここでは OpenCV の画像に BGR フォーマットを使用することとし、チャンネル数には 3 を設定しています。

3.5.2. `ProcessLoop()`関数の代わりに `ImageHandling()`関数をコールするよう変更

プログラム開始時にコマンドプロンプト上で処理開始のためのキー入力を行うと、`OpenCV` の画像表示ウィンドウを表示してカメラから受信した画像を画像表示ウィンドウに表示するようコードを変更します。

`ImageHandling()`は関数起動時に画像表示ウィンドウを表示し、Escape キーが押されると画像表示ウィンドウを閉じて `ImageHandling()`を終了するように `3.6` 節で改造します。`ImageHandling()`が異常終了したときにエラーコメントが確認できるよう、キー入力待ちの処理も追加してください。

`ImageHandling()`が正常終了したときには確認のためのキー入力を待たずに Stream_FreerunLockBuffer を終了させるようにするために、`tmain()`末尾のキー入力待ち処理をコメントアウトしてください。

3.5.3. OpenCV の画像バッファに割り付けたメモリを解放するコードの追加

`ImageHandling()`を呼ぶコードの後に OpenCV 画像バッファのメモリを開放するコードを追加してください。

```
int _tmain(int argc, _TCHAR* argv[])
{
    CAM_API_STATUS  uiStatus     = CAM_API_STS_SUCCESS;
    bool8_t         bStatus      = true;
    uint32_t        uiNum        = 0;
    uint32_t        uiImgBufSize = 0;       // Size of image buffer.

    printf("Stream_FreerunLockBuffer sample application for TeliCamSDK and OpenCV.\n\n");
    printf("Copyrights (C) 2014 - 2018 TOSHIBA TELI CORPORATION. All rights reserved.\n\n");
    printf("Press any key to start...\n");
    _getch();

    do
    {
        // API initialization.
        uiStatus = Sys_Initialize();
        if (uiStatus != CAM_API_STS_SUCCESS)
        {
            printf("Failed Sys_Initialize, Status = 0x%08X.\n", uiStatus);
            break;
        }

        // Get number of camera.
        uiStatus = Sys_GetNumOfCameras(&uiNum);
        if (uiStatus != CAM_API_STS_SUCCESS)
        {
            printf("Failed Sys_GetNumOfCameras, Status = 0x%08X.\n", uiStatus);
            break;
        }
        if (uiNum == 0)
        {
            printf(" No cameras found.\n");
            break;
        }

        // Open camera that is detected first, in this sample code.
        uiStatus = Cam_Open(0, &s_hCam);
        if (uiStatus != CAM_API_STS_SUCCESS)
        {
            printf("Failed Cam_Open, Status = 0x%08X.\n", uiStatus);
            break;
        }

        // Create completion event object for stream.
        s_hStrmCmpEvt = CreateEvent(NULL, FALSE, FALSE, NULL);
        if (s_hStrmCmpEvt == NULL)
        {
            printf("Failed CreateEvent.\n");
            break;
        }

        // Open stream.
        uiStatus = Strm_OpenSimple(s_hCam, &s_hStrm, &uiImgBufSize, s_hStrmCmpEvt);
        if (uiStatus != CAM_API_STS_SUCCESS)
        {
            printf("Failed Strm_OpenSimple, Status = 0x%08X.\n", uiStatus);
            break;
        }

        // Allocate memory to OpenCV image object.
        //   Gets width and height of image.
        uint32_t uiWidth, uiHeight;

        uiStatus = GetCamWidth(s_hCam, &uiWidth);
        if (uiStatus != CAM_API_STS_SUCCESS)
        {
            printf("Failed GetCamWidth, Status = 0x%08X.\n", uiStatus);
            break;
        }
```

3.5.1

```
        uiStatus = GetCamHeight(s_hCam, &uiHeight);
        if (uiStatus != CAM_API_STS_SUCCESS)
        {
            printf("Failed GetCamHeight, Status = 0x%08X.\n", uiStatus);
            break;
        }

        //    Allocate memory to OpenCV image object.
#ifdef USE_IPL_IMAGE
        s_pIplImage = ::cvCreateImage(::cvSize(uiWidth, uiHeight), IPL_DEPTH_8U, 3);
#else
        s_openCvImage = cv::Mat(cv::Size(uiWidth, uiHeight), CV_8UC3, cv::Scalar::all(0));
#endif

       // Set free-run mode.
       // Set TriggerMode false.
       uiStatus = SetCamTriggerMode(s_hCam, false);
       if (uiStatus != CAM_API_STS_SUCCESS)
       {
           printf("Failed SetCamTriggerMode, Status = 0x%08X.\n", uiStatus);
           break;
       }

       // Receive images.
        // bStatus = ProcessLoop();
       //if (!bStatus)
       //{
       //    break;
       //}

       // Receive images.
        if (!ImageHandling())
        {
            printf("Press any key to exit...\n");
            _getch();
        }

    } while (0);

    // Close stream.
    if (s_hStrm != (CAM_STRM_HANDLE)NULL)
    {
        Strm_Close(s_hStrm);
        s_hStrm = (CAM_STRM_HANDLE)NULL;
    }

    if (s_hStrmCmpEvt != NULL)
    {
        CloseHandle(s_hStrmCmpEvt);
        s_hStrmCmpEvt = NULL;
    }

    // Release image buffer of OpenCV image object.
#ifdef USE_IPL_IMAGE
    if (s_pIplImage != NULL)
    {
        ::cvReleaseImage(&s_pIplImage);
        s_pIplImage = NULL;
    }
#else
    // Do Nothing
#endif

    // Close camera.
    if (s_hCam != (CAM_HANDLE)NULL)
    {
        Cam_Close(s_hCam);
        s_hCam = (CAM_HANDLE)NULL;
    }
```

3.5.1

3.5.2

3.5.3


```
    // Terminate system.
    Sys_Terminate();
    //printf("Finished.\n\n");
    //printf("Press any key to exit...\n");
    //_getch();

    return 0;
}
```

3.6. ImageHandling 関数の変更

ImageHandling()関数では OpenCV の画像表示ウィンドウを表示・解放する処理、受信した画像データを画像表示ウィンドウに表示する処理、画像表示ウィンドウをクローズするためのキー入力監視処理を追加してください。

3.6.1. OpenCV の画像表示ウィンドウを作成するコードの追加

ImageHandling ()関数の最初の部分に OpenCV の画像表示ウィンドウを作成するコードを追加してください。キー入力監視処理で使用する変数 key もここで宣言してください。

3.6.2. キー入力チェック処理のコメントアウト

while()ループの先頭にある画像取得終了指示受け付け用のキー入力チェック処理は不要になりますのでコメントアウトしてください。

3.6.3. 取り込んだ画像データを OpenCV の画像オブジェクトにコピーするコードの追加。

カメラユーティリティ関数 ConvImage()でカメラから受信した画像データを BGR フォーマットに変換して OpenCV の画像オブジェクトに設定するコードを追加してください。受信した画像データとその縦横画像サイズと PixelFormat は Strm_LockBuffer()で取得した画像情報構造体 imageInfo から取得します。

この例では ConvImage() 関数の第２引数にカメラから受信した画像情報の PixelFormat メンバ変数を指定していますが、カメラの機種とファームウェアバージョンによっては ConvImage() 関数がエラーを返したり、変換後の画像が不適切な画像になる場合があります。このようなときは画像情報の PixelFormat メンバ変数の代わりに GetCamPixelFormat()関数で取得した PixelFormat 値を指定してください。

3.6.4. OpenCV の画像オブジェクト内の画像を OpenCV 画面に表示するコードの追加

::cvShowImage()または cv::imshow()で画像を OpenCV 画面に表示するよう指示するコードを追加します。waitKey() 関数がコールされたタイミングで OpenCV ウィンドウの描画処理が実行されます。

3.6.5. While ループから抜けるためのコードの追加

エスケープキーが押されたら while 文から抜けるコードを追加してください。

3.6.6. 不要となった Print 文のコメントアウト

受信画像の先頭 8byte の表示は行わないようにします。

3.6.7. OpenCV の画像表示ウィンドウを解放するコードの追加

ImageHandling ()関数の最後の部分に OpenCV の画像表示ウィンドウを解放するコードを追加してください。

#5400-0444


```
bool8_t ImageHandling()
{
    CAM_API_STATUS  uiStatus      = CAM_API_STS_SUCCESS;
    uint32_t        uiRes;
    uint32_t        uiBufferIndex = 0;
    uint8_t        *pImageBuf     = NULL;
    bool8_t         bStatus       = true;
    CAM_IMAGE_INFO  imageInfo;

    // Declare key value variable for getting pressed key.
    int             key;
    // Create OpenCV window for showing image.
    cv::namedWindow("OpenCV Test", CV_WINDOW_AUTOSIZE);

    // Start stream.
    uiStatus = Strm_Start(s_hStrm);
    if (uiStatus != CAM_API_STS_SUCCESS)
    {
         bStatus = false;
         printf("Failed Strm_Start, Status = 0x%08X.\n", uiStatus);
        return false;
    }

    while (_kbhit())
    {
         _getch();
    }

    while (1)
    {
         //if (_kbhit())
         //{
         //    _getch();
         //    break;
         //}

         // Wait for receiving image completion event.
         uiRes = WaitForSingleObject(s_hStrmCmpEvt, 2000);
         if (uiRes != WAIT_OBJECT_0)
         {
              bStatus = false;
              printf("Timeout WaitForSingleObject.\n");
              break;
         }

         // Get current ring buffer index.
         uiStatus = Strm_GetCurrentBufferIndex(s_hStrm, &uiBufferIndex);
         if (uiStatus != CAM_API_STS_SUCCESS)
         {
              bStatus = false;
              printf("Failed Strm_GetCurrentBufferIndex, Status = 0x%08X.\n", uiStatus);
              break;
         }

         // Lock the ring buffer pointer.
         uiStatus = Strm_LockBuffer(s_hStrm, uiBufferIndex, &imageInfo);
         if (uiStatus != CAM_API_STS_SUCCESS)
         {
              bStatus = false;
              printf("Failed Strm_LockBuffer, Status = 0x%08X.\n", uiStatus);
              break;
         }

#ifdef USE_IPL_IMAGE
         if (PXL_FMT_Mono8 == s_pixelFormat)
         {
              // Copy received image data to OpenCV image object.
              rsize_t size = imageInfo.uiSizeX * imageInfo.uiSizeY;
              memcpy_s(s_pIplImage->imageData, size, imageInfo.pvBuf, size);
         }
```

3.6.1.

3.6.2.

3.6.3.


```
        else
        {
            // Copy converted image data to OpenCV image object.
            uiStatus = ConvImage(DST_FMT_BGR24, imageInfo.uiPixelFormat, true,
                s_pIplImage->imageData, imageInfo.pvBuf,
                imageInfo.uiSizeX, imageInfo.uiSizeY);
            if (uiStatus != CAM_API_STS_SUCCESS)
            {
                //bStatus = false;
                printf("Failed ConvImage, Status = 0x%08X.\n", uiStatus);
                break;
            }
        }

        // Show image in OpenCV window.
        ::cvShowImage("OpenCV Test", s_pIplImage);
#else
        if (PXL_FMT_Mono8 == s_pixelFormat)
        {
            // Copy received image data to OpenCV image object.
            rsize_t size = imageInfo.uiSizeX * imageInfo.uiSizeY;
            memcpy_s(s_openCvImage.data, size, imageInfo.pvBuf, size);
        }
        else
        {
            // Copy converted image data to OpenCV image object.
            uiStatus = ConvImage(DST_FMT_BGR24, imageInfo.uiPixelFormat, true,
                                 s_openCvImage.data, imageInfo.pvBuf,
                                 imageInfo.uiSizeX, imageInfo.uiSizeY);
            if (uiStatus != CAM_API_STS_SUCCESS)
            {
                bStatus = false;
                printf("Failed ConvImage, Status = 0x%08X.\n", uiStatus);
                break;
            }
        }

        // Show image in OpenCV window.
        cv::imshow("OpenCV Test", s_openCvImage);
#endif

        // Add code for checking that Escape key is pressed.
        key = cv::waitKey(10);
        if (key == 0x1b)    // [ESC] key
        {
            break;
        }

       //// Display the first 8 pixel values(8 bit color).
       //pImageBuf = (uint8_t*)imageInfo.pvBuf;
       //printf("  Image :");
       //for (int i = 0; i < 8; i++)
       //{
       //    printf(" %02X ", pImageBuf[i]);
       //}
       //printf("...\n");

        // Unlock the ring buffer pointer.
        uiStatus = Strm_UnlockBuffer(s_hStrm, uiBufferIndex);
        if (uiStatus != CAM_API_STS_SUCCESS)
        {
            bStatus = false;
            printf("Failed Strm_UnlockBuffer, Status = 0x%08X.\n", uiStatus);
            break;
        }
    }

    // Destroy OpenCV image window.
    cv::destroyWindow("OpenCV Test");
```

3.6.3.
3.6.4.
3.6.3
3.6.4.
3.6.5.
3.6.6.
3.6.7.

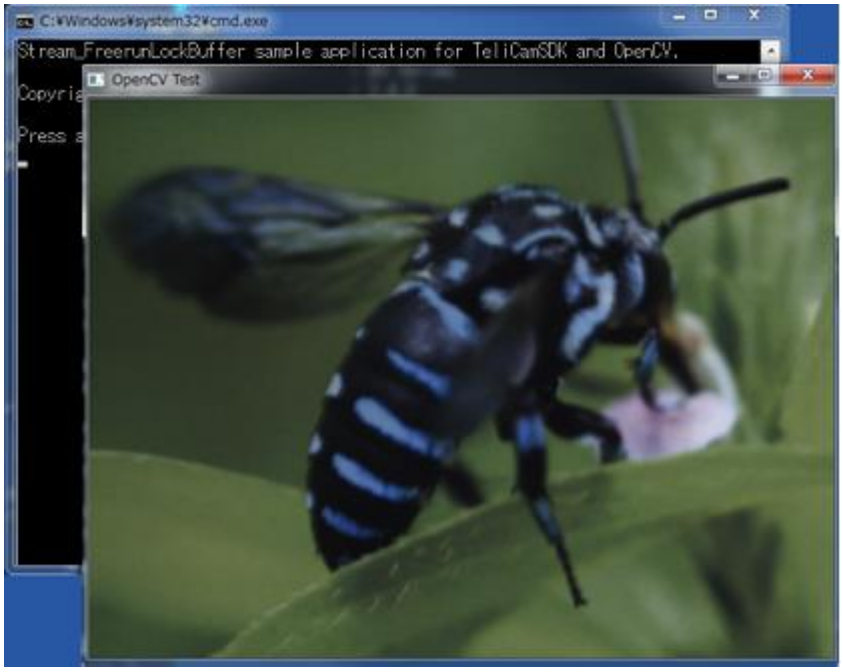
```
    //Stop stream.
    uiStatus = Strm_Stop(s_hStrm);
    if (uiStatus != CAM_API_STS_SUCCESS)
    {
        bStatus = false;
        printf("Failed Strm_Stop, Status = 0x%08X.\n", uiStatus);
    }

    Sleep(100);

    return bStatus;
}
```

3.7. ProcessLoop()処理の削除

オリジナルの ProcessLoop()関数は不要になりますので削除してください。

3.8. 改造した Stream_FreerunLockBuffer の動作確認

コードの改造が終わったら、「デバッグ」→「デバッグなしで実行」を選択し、カメラで撮像した画像が OpenCV の画面に表示されることを確認してください。



OpenCV の DLL が見つからないというエラーメッセージが表示された場合は、OpenCV の DLL フォルダのパスが環境変数 PATH に設定されていない可能性があります。

環境変数 PATH に、使用する Visual Studio のバージョンに合った dll が保存されているフォルダを追加してください。

	32bit (x86)	64bit (x64)
Visual Studio 2015	c:\opencv\build\x86\vc14\bin	c:\opencv\build\x64\vc14\bin
Visual Studio 2017	c:\opencv\build\x86\vc15\bin	c:\opencv\build\x64\vc15\bin

環境変数 PATH は以下の手順で編集できます。

- 「コントロールパネル」→「システム」→「システムの詳細設定」でシステムのプロパティを開く。
- [環境変数] ボタンをクリックする。
  環境変数画面が表示されます。
- 「システム環境変数」の表を下にスクロールさせて変数 Path を選択し、[編集]ボタンをクリックする。
- システム変数の編集画面で変数値(V)に表示される値にセミコロン区切りでフォルダ名を追加する。
- ［OK］ボタンをクリックしてシステム変数の編集画面を閉じ、環境変数画面などの今までに開いた画面も［OK］ボタンクリックですべて閉じる。

3.9. 改造後の Stream_FreerunLockBufer.cpp

改造後の Stream_FreerunLockBuffer.cpp を以下に示します。

このコードでは前述の変更の他に、カメラから出力される画像フォーマットが Mono8 のときのみ、OpenCV の画像チャンネル数を 1 チャンネルにし、ConvImage()関数を使用しないで Strm_LockBuffer()で取得した画像データを直接 OpenCV の画像メモリにコピーするように変更しています。

```
#include <windows.h>
#include <tchar.h>
#include <conio.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#include "opencv2/opencv.hpp"

#include "TeliCamApi.h"
#include "TeliCamUtl.h"

using namespace Teli;

/*****************************************************************************/
/* Prototype declares                                                        */
/*****************************************************************************/

bool8_t ImageHandling();

/*****************************************************************************/
/* Static Values                                                             */
/*****************************************************************************/

// Handles
static CAM_HANDLE       s_hCam       = (CAM_HANDLE)NULL;             // Camera handle.
static CAM_STRM_HANDLE  s_hStrm      = (CAM_STRM_HANDLE)NULL;        // Stream handle.
static HANDLE           s_hStrmCmpEvt = NULL;                       //event object for stream.
static CAM_PIXEL_FORMAT s_pixelFormat;

//#define USE_IPL_IMAGE
#ifdef USE_IPL_IMAGE
static ::IplImage        *s_pIplImage;
#else
static cv::Mat           s_openCvImage;
#endif

/*****************************************************************************/
/* Functions                                                                 */
/*****************************************************************************/
int _tmain(int argc, _TCHAR* argv[])
{
    CAM_API_STATUS  uiStatus     = CAM_API_STS_SUCCESS;
    bool8_t         bStatus      = true;
    uint32_t        uiNum        = 0;
    uint32_t        uiImgBufSize = 0;       // Size of image buffer.

    printf("Stream_FreerunLockBuffer sample application for TeliCamSDK and OpenCV.\n\n");
    printf("Copyrights (C) 2014 - 2018 TOSHIBA TELI CORPORATION. All rights reserved.\n\n");
    printf("Press any key to start...\n");
    _getch();

    do
    {
        // API initialization.
        uiStatus = Sys_Initialize();
        if (uiStatus != CAM_API_STS_SUCCESS)
        {
            printf("Failed Sys_Initialize, Status = 0x%08X.\n", uiStatus);
```

```
            break;
        }

        // Get number of camera.
        uiStatus = Sys_GetNumOfCameras(&uiNum);
        if (uiStatus != CAM_API_STS_SUCCESS)
        {
            printf("Failed Sys_GetNumOfCameras, Status = 0x%08X.\n", uiStatus);
            break;
        }
        if (uiNum == 0)
        {
            printf(" No cameras found.\n");
            break;
        }

        // Open camera that is detected first, in this sample code.
        uiStatus = Cam_Open(0, &s_hCam);
        if (uiStatus != CAM_API_STS_SUCCESS)
        {
            printf("Failed Cam_Open, Status = 0x%08X.\n", uiStatus);
            break;
        }

        // Create completion event object for stream.
        s_hStrmCmpEvt = CreateEvent(NULL, FALSE, FALSE, NULL);
        if (s_hStrmCmpEvt == NULL)
        {
            printf("Failed CreateEvent.\n");
            break;
        }

        // Open stream.
        uiStatus = Strm_OpenSimple(s_hCam, &s_hStrm, &uiImgBufSize, s_hStrmCmpEvt);
        if (uiStatus != CAM_API_STS_SUCCESS)
        {
            printf("Failed Strm_OpenSimple, Status = 0x%08X.\n", uiStatus);
            break;
        }

         // Allocate memory to OpenCV image object.
         //   Gets width, height and PixelFormat of image.
         uint32_t uiWidth, uiHeight;

         uiStatus = GetCamWidth(s_hCam, &uiWidth);
         if (uiStatus != CAM_API_STS_SUCCESS)
         {
             printf("Failed GetCamWidth, Status = 0x%08X.\n", uiStatus);
             break;
         }

         uiStatus = GetCamHeight(s_hCam, &uiHeight);
         if (uiStatus != CAM_API_STS_SUCCESS)
         {
             printf("Failed GetCamHeight, Status = 0x%08X.\n", uiStatus);
             break;
         }

         uiStatus = GetCamPixelFormat(s_hCam, &s_pixelFormat);
         if (uiStatus != CAM_API_STS_SUCCESS)
         {
             printf("Failed GetCamPixelFormat, Status = 0x%08X.\n", uiStatus);
             break;
         }

         //    Allocate memory to OpenCV image object.
#ifdef USE_IPL_IMAGE
         if (PXL_FMT_Mono8 == s_pixelFormat)
         {
             s_pIplImage = ::cvCreateImage(::cvSize(uiWidth, uiHeight), IPL_DEPTH_8U, 1);
```

```
        }
        else
        {
            s_pIplImage = ::cvCreateImage(::cvSize(uiWidth, uiHeight), IPL_DEPTH_8U, 3);
        }

#else
        if (PXL_FMT_Mono8 == s_pixelFormat)
        {
            s_openCvImage = cv::Mat(cv::Size(uiWidth, uiHeight), CV_8UC1, cv::Scalar::all(0));
        }
        else
        {
            s_openCvImage = cv::Mat(cv::Size(uiWidth, uiHeight), CV_8UC3, cv::Scalar::all(0));
        }
#endif
       // Set free-run mode.
       // Set TriggerMode false.
       uiStatus = SetCamTriggerMode(s_hCam, false);
       if (uiStatus != CAM_API_STS_SUCCESS)
       {
           printf("Failed SetCamTriggerMode, Status = 0x%08X.\n", uiStatus);
           break;
       }

       // Receive images.
        if (!ImageHandling())
        {
            printf("Press any key to exit...\n");
            _getch();
        }
    } while (0);

    // Close stream.
    if (s_hStrm != (CAM_STRM_HANDLE)NULL)
    {
        Strm_Close(s_hStrm);
        s_hStrm = (CAM_STRM_HANDLE)NULL;
    }

    if (s_hStrmCmpEvt != NULL)
    {
        CloseHandle(s_hStrmCmpEvt);
        s_hStrmCmpEvt = NULL;
    }

    // Release image buffer of OpenCV image object.
#ifdef USE_IPL_IMAGE
    if (s_pIplImage != NULL)
    {
        ::cvReleaseImage(&s_pIplImage);
        s_pIplImage = NULL;
    }
#else
    // Do Nothing
#endif

    // Close camera.
    if (s_hCam != (CAM_HANDLE)NULL)
    {
        Cam_Close(s_hCam);
        s_hCam = (CAM_HANDLE)NULL;
    }

    // Terminate system.
    Sys_Terminate();

    return 0;
}
```



```
////////////////////////////////////////////////////////
//
bool8_t ImageHandling()
{
    CAM_API_STATUS  uiStatus     = CAM_API_STS_SUCCESS;
    uint32_t        uiRes;
    uint32_t        uiBufferIndex = 0;
    uint8_t        *pImageBuf    = NULL;
    bool8_t         bStatus      = true;
    CAM_IMAGE_INFO  imageInfo;

    // Declare key value variable for getting pressed key.
    int             key;
    // Create OpenCV window for showing image.
    cv::namedWindow("OpenCV Test", CV_WINDOW_AUTOSIZE);

    // Start stream.
    uiStatus = Strm_Start(s_hStrm);
    if (uiStatus != CAM_API_STS_SUCCESS)
    {
        bStatus = false;
        printf("Failed Strm_Start, Status = 0x%08X.\n", uiStatus);
       return false;
    }

    while (_kbhit())
    {
        _getch();
    }

    while (1)
    {
        // Wait for receiving image completion event.
        uiRes = WaitForSingleObject(s_hStrmCmpEvt, 2000);
        if (uiRes != WAIT_OBJECT_0)
        {
            bStatus = false;
            printf("Timeout WaitForSingleObject.\n");
            break;
        }

        // Get current ring buffer index.
        uiStatus = Strm_GetCurrentBufferIndex(s_hStrm, &uiBufferIndex);
        if (uiStatus != CAM_API_STS_SUCCESS)
        {
            bStatus = false;
            printf("Failed Strm_GetCurrentBufferIndex, Status = 0x%08X.\n", uiStatus);
            break;
        }

        // Lock the ring buffer pointer.
        uiStatus = Strm_LockBuffer(s_hStrm, uiBufferIndex, &imageInfo);
        if (uiStatus != CAM_API_STS_SUCCESS)
        {
            bStatus = false;
            printf("Failed Strm_LockBuffer, Status = 0x%08X.\n", uiStatus);
            break;
        }

#ifdef USE_IPL_IMAGE
        if (PXL_FMT_Mono8 == s_pixelFormat)
        {
            // Copy received image data to OpenCV image object.
            rsize_t size = imageInfo.uiSizeX * imageInfo.uiSizeY;
            memcpy_s(s_pIplImage->imageData, size, imageInfo.pvBuf, size);
        }
        else
        {
            // Copy converted image data to OpenCV image object.
            uiStatus = ConvImage(DST_FMT_BGR24, imageInfo.uiPixelFormat, true,
```

```
                s_pIplImage->imageData, imageInfo.pvBuf,
                imageInfo.uiSizeX, imageInfo.uiSizeY);
            if (uiStatus != CAM_API_STS_SUCCESS)
            {
                //bStatus = false;
                printf("Failed ConvImage, Status = 0x%08X.\n", uiStatus);
                break;
            }
        }

        // Show image in OpenCV window.
        ::cvShowImage("OpenCV Test", s_pIplImage);
#else
        if (PXL_FMT_Mono8 == s_pixelFormat)
        {
            // Copy received image data to OpenCV image object.
            rsize_t size = imageInfo.uiSizeX * imageInfo.uiSizeY;
            memcpy_s(s_openCvImage.data, size, imageInfo.pvBuf, size);
        }
        else
        {
            // Copy converted image data to OpenCV image object.
            uiStatus = ConvImage(DST_FMT_BGR24, imageInfo.uiPixelFormat, true,
                                 s_openCvImage.data, imageInfo.pvBuf,
                                 imageInfo.uiSizeX, imageInfo.uiSizeY);
            if (uiStatus != CAM_API_STS_SUCCESS)
            {
                bStatus = false;
                printf("Failed ConvImage, Status = 0x%08X.\n", uiStatus);
                break;
            }
        }
        // Show image in OpenCV window.
        cv::imshow("OpenCV Test", s_openCvImage);
#endif

        // Add code for checking that Escape key is pressed.
        key = cv::waitKey(10);
        if (key == 0x1b)    // [ESC] key
        {
            break;
        }

       // Unlock the ring buffer pointer.
       uiStatus = Strm_UnlockBuffer(s_hStrm, uiBufferIndex);
       if (uiStatus != CAM_API_STS_SUCCESS)
       {
           bStatus = false;
           printf("Failed Strm_UnlockBuffer, Status = 0x%08X.\n", uiStatus);
           break;
       }
    }

    // Destroy OpenCV image window.
    cv::destroyWindow("OpenCV Test");

    //Stop stream.
    uiStatus = Strm_Stop(s_hStrm);
    if (uiStatus != CAM_API_STS_SUCCESS)
    {
        bStatus = false;
        printf("Failed Strm_Stop, Status = 0x%08X.\n", uiStatus);
    }

    Sleep(100);

    return bStatus;
}
```


付録 1. 他のバージョンの OpenCV を使用する場合

バージョン 2 系または 3.4.3 以外の 3 系の OpenCV をご使用の場合、Visual Studio 用のバイナリが提供されている OpenCV バージョンであれば、参照する OpenCV ライブラリの設定（3.2 節）以外は 3 章に記載している通りに編集し、参照する OpenCV ライブラリのみ本章記載のように設定すれば Stream_FreerunLockBuffer を動作させることができます。

付録1.1.「追加のライブラリディレクトリ」の編集

OpenCV のバージョンによって提供されているバイナリの Visual Studio バージョンが変わります。使用する lib ファイルが入っているフォルダを「追加のライブラリディレクトリ」に追加してください。

	32bit (x86)	64bit (x64)
Visual Studio 2010	c:/opencv/build/x86/vc10/lib	c:/opencv/build/x64/vc10/lib
Visual Studio 2012	c:/opencv/build/x86/vc11/lib	c:/opencv/build/x64/vc11/lib
Visual Studio 2013	c:/opencv/build/x86/vc12/lib	c:/opencv/build/x64/vc12/lib
Visual Studio 2015	c:/opencv/build/x86/vc14/lib	c:/opencv/build/x64/vc14/lib
Visual Studio 2017	c:/opencv/build/x86/vc15/lib	c:/opencv/build/x64/vc15/lib

使用する Visual Studio のバージョン用の OpenCV ライブラリが提供されていない場合、他のバージョンの Visual Studio 用の OpenCV ライブラリを使用することも可能です。

但し、作成したアプリケーションの実行には、コンパイルに使用しているバージョンの Visual Studio 用 VC++ランタイムと OpenCV ライブラリのコンパイルに使用されたバージョンの VC++ランタイムの両方が必要になることにご留意ください。

付録1.2.「追加の依存ファイル」の編集

付録1.2.1 OpenCV 3.x の場合

「opencv_world**xxx**.lib」の「**xxx**」をバージョン名から”.”を取り除いた文字列に置き換えたファイル名を「追加の依存ファイル」に追加してください。

付録1.2.2 OpenCV 2.x の場合

「opencv_core**xxx**.lib」「opencv_highgui**xxx**.lib」「opencv_imgproc**xxx**.lib」の 3lib ファイルを「追加の依存ファイル」に追加してください。各ファイル名の「**xxx**」の部分はバージョン名から”.”を取り除いた文字列に置き換えてください。

End of document in Japanese

#5400-0444

4. Others

4.1. Revision History

Date	Version	Description
2018/11/15	1.0.0	Created the initial version

4.2. Disclaimer

The disclaimer of this document including example code is described in “License Agreement TeliCamSDK Eng.pdf” in TeliCamSDK installation folder.
Make sure to read this Agreement carefully before using it.
Refer to TeliCamSDK installation folder/Documents/License folder

4.3. 8.3. License

Microsoft and Visual C++ are trade mark or registered trade mark of Microsoft Corporation
GigE Vision™ is standard by AIA(Automated Imaging Association).
USB3 Vision™ is standard by AIA(Automated Imaging Association)
GenICam™ is registered trade mark of EMVA(European Machine Vision association)
Other product names in this document are trade mark or registered trade mark of these companies.

