FigSpec-2X-SDK C# Development Guide

Revision Record:

Update Version	Revised content
2024-09-05	Initial version

1 Summary

The FS-2X series hyperspectral camera can capture a plane, with a certain spatial resolution, multiple spectral channels for simultaneous data acquisition.

For example, the FS-23 acquisition image is composed of 1920 lines, wherein the spatial resolution of each line is 1920, the number of spectral channels is 1200, and the spectral range is 400-1000nm. The specific meaning is: FS-23 can collect a face, which consists of 1920,Each line is divided into 1920 points in space, and 1200 spectral channels

are performed for each point while data acquisition is performed. Each spectral channel corresponds to a different wavelength of light.

Note: Different models of FS-2X series line scan hyperspectral cameras differ in spatial resolution, number of spectral channels and spectral range.

The FigSpec hyperspectral camera (FS-2X series) SDK is an independent component used to control the camera. It supports functions such as obtaining real-time hyperspectral images and configuring parameters. The FigSpec hyperspectral camera (FS-2X series) SDK is compatible with USB3 Vision protocol and currently supports Windows system.The basic functions of the SDK for camera control include parameter setting and image acquisition:

- Parameter setting: Obtain and set all parameters of the camera, which can be realized by using the common configuration interface.
- Hyperspectral Image Acquisition: when the camera sends hyperspectral images to the host, the hyperspectral images are stored in the SDK, which provides a callback method to acquire hyperspectral images.

2 Description of the camera model, operating system, and operating environment supported by the SDK

2.1 Camera models supported by the SDK

Suitable for: FS-22, FS-23, FS-24, FS-27

2.2 SDK support System description

Windows 7 Service Pack 1 (SP1)、Windows 8、Windows 10、Windows 11

2.3 SDK support System description

. NET Framework 4.6.1 and above, all code demonstrations in this document are described in C.

3 File Directory Introduction

In the FigSpec-Camera-SDK folder, include the SDK secondary development dependency files. The directory structure is as follows:

- Folder: Documents:
  - SDK development guide
- Folders: Libraries
  - Camera-related class library files; Note: Developers need to copy this file to the running directory of their own programs.

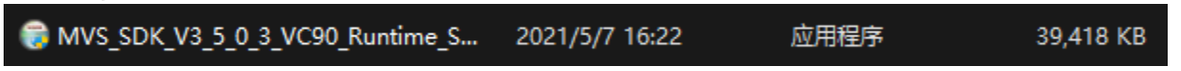
- Folder: Drivers

  The drivers to be installed are placed in the folder named after the camera model. Please install the corresponding camera driver according to the camera model. Among them, the model is FS-24 and FS-27 cameras do not need to install camera drivers.
  - FS-22: Indicates the content directory that needs to be installed for cameras with model F-22
  - FS-23: Indicates the content directory that needs to be installed for cameras with model F-23
- Folder: Samples (sample code, including the following example)
  - Example1 folder: Example 1 (the camera is working in "line scan mode", the operation mode is ALL Mode, spectral dimension and spatial dimension 1 are combined to obtain 10 frames of hyperspectral data, and the hyperspectral data is converted into reflectivity data, and the data is saved as a hyperspectral image file)
  - Example2 folder: Example 2 (the camera works in "line scan mode" and the operation mode is BANK mode, obtaining 10 frames of hyperspectral data)
  - Example3 folder: Example 3 (the camera works in "imaging hyperspectral mode", the operation mode is ALL Mode, spectral dimension and spatial dimension 1 are combined to obtain a hyperspectral image, which is saved as a hyperspectral image file)
  - Example4 folder: Example 4 (1 Hyperspectral image is obtained when the camera is working in "imaging hyperspectral mode" and the operation mode is BANK mode)
  - Example5 folder: Example 5 (get 100 preview pictures and save them as files)

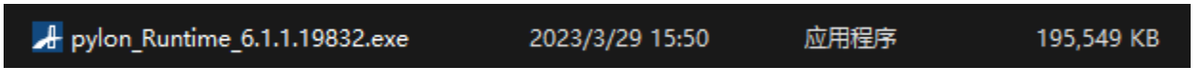
4 Device driver installation

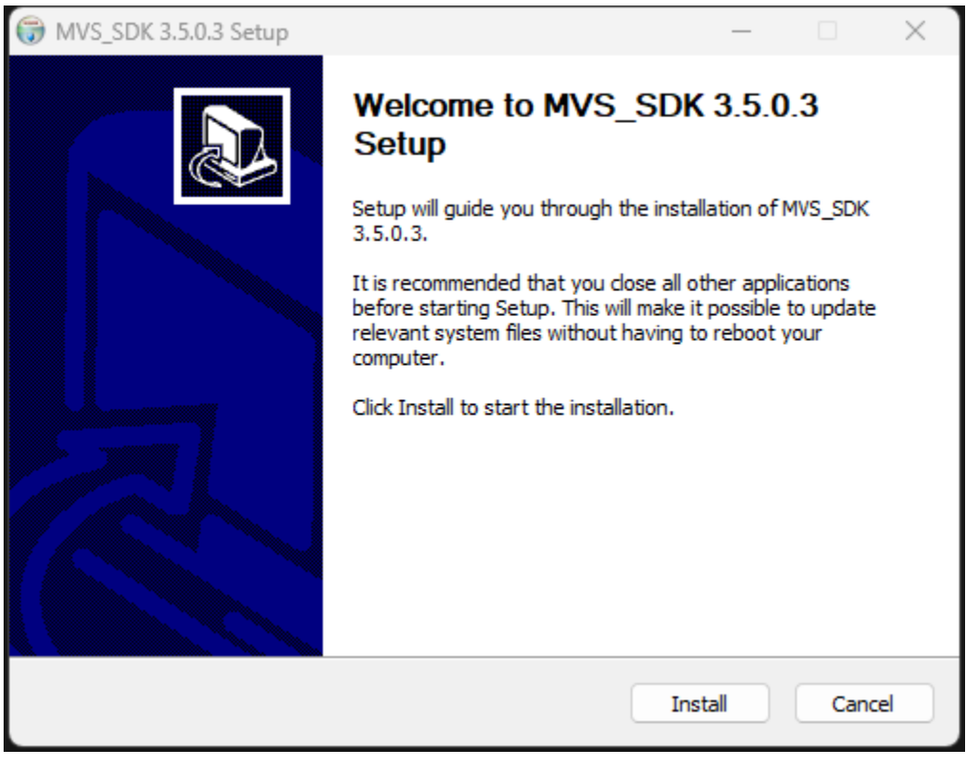
4.1 Driver installation for camera model FS22

The driver files that need to be installed are respectively named: Driver Packages. In the FigSpec-Camera-SDK/Driver Directory



Installation steps:



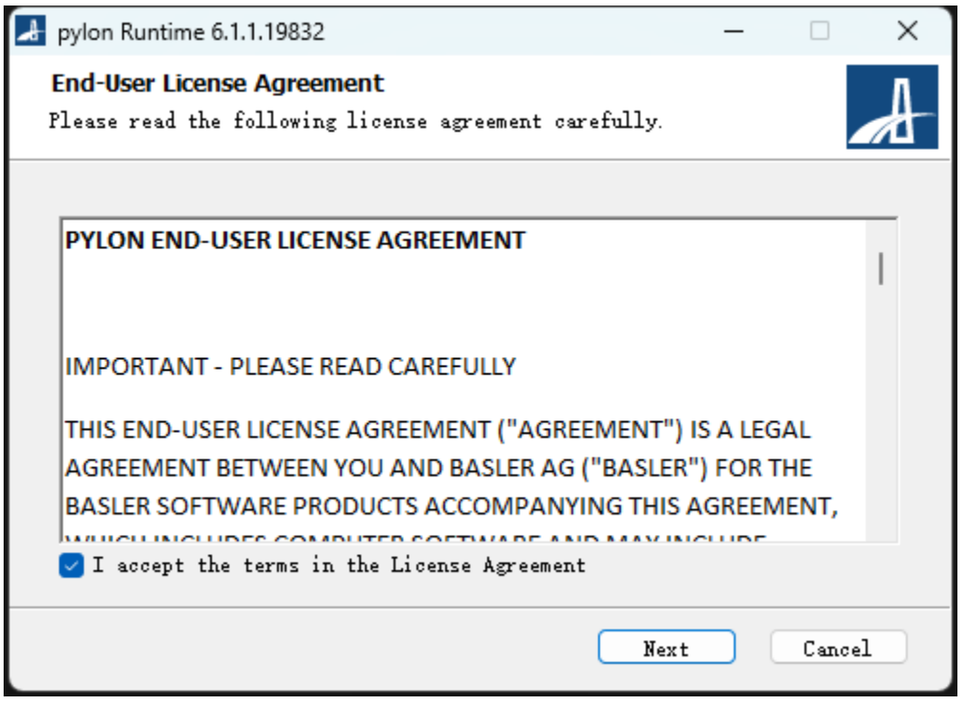


1. The mouse double-click "" The driver package.
2. Click the "Install" button to Install.
3. Wait for the program installation to complete.

4.2 Driver Installation for Camera Model FS23

The driver files that need to be installed are the driver packages named "pylon_Runtime_6.1.1.19832.exe. In the FigSpec-Camera-SDK/Driver Directory

Installation steps:



1. Double-click the driver package of "pylon_Runtime_6.1.1.19832.exe.
2. Check the "I accept the terms in the License agreement" box and click the "Next" button to proceed to the Next step.
3. Select the installation path and click the "Install" button to Install.
4. Wait for the program installation to complete.

4.3 Driver installation for camera model FS24, FS27

This model of camera does not require the installation of additional drivers on Win8 and above systems.

5 External trigger (external IO) wiring instructions

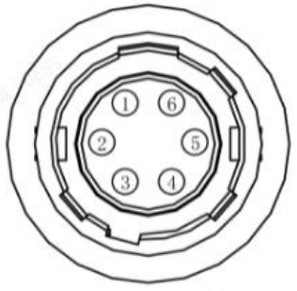
5.1 External trigger wiring instructions for camera model FS22

The pin signals corresponding to the camera's external IO interface are defined in the following table:

6-pin P7 interface diagram

Pin Signal Definition

Pin	Signal	I/O signal source	Explain
1	DC_PWR	--	Camera light source
2	OPTO_IN	LINE 0+	Optocoupler Isolated Input
3	GPIO	LINE 2+	Configurable input or output
4	OPTO_OUT	LINE 1+	Optocoupler isolated output
5	OPTO_GND	LINE 0-/1-	Optocoupler isolation signal ground

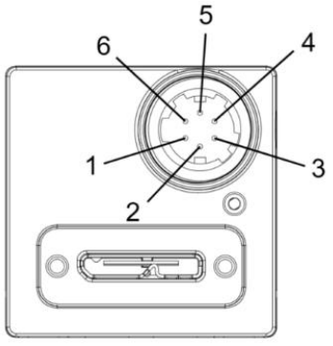


6	GND	LINE 2-	Camera power ground

The signal trigger ground line is pin 5, and the signal trigger line is pin 2. Among them, the working voltage of pin 2 is 3.3V-24V and the current is 5-25mA.

5.2 Description of external trigger wiring for camera model FS23

The pin signals corresponding to the camera's external IO interface are defined in the following table:



Pin	Line	Function
1	Line 3	General I/O(GPIO) line
2	Line 1	Opto-coupled I/O input line
3	Line 4	General I/O(GPIO) line
4	Line 2	Opto-coupled I/O output line
5	-	Photoelectric coupling I/O line grounding
6	-	General purpose I/O(GPIO) line grounding

The signal trigger ground wire is pin 5, and the signal trigger wire is pin 2. The working voltage of pin 2 is 3.3V-24V and the current is 5-15mA.

5.3 Description of external trigger wiring for camera model FS24, FS27

The pin signals corresponding to the camera's external IO interface are defined in the following table:

Color	Pin	Signal	Signal Description
Red	1	DIR_GPIO	Non-isolated signal (software configurable input/output)(line2)
White	2	OPTO_GND	Optocoupler isolation signal ground
Blue	3	OPTO_OUT	Optocoupler Isolated Output Signal (line1)

Green	4	OPTO_IN	Optocoupler isolation input signal (line0)
Black	5	GND	Non-isolated signal and power ground
Yellow	6	5V	5V DC power input

The signal trigger ground line is pin 2, and the signal trigger line is pin 4. The working voltage of pin 4 is 3.3V-24V, and the current is 5-30mA.

6 Introduction of camera parameters

6.1 What is Hyperspectral Image Data?

In an image sampled by an ordinary camera, each point is usually represented by three RGB values, which respectively represent the size of the red, green, and blue signals of a point. In the spectral image, each point is represented by the signal strength of many wavelengths. These wavelengths can have a wider spectral range, such as visible light covering not only 400-700nm,It can also be extended to near infrared light from 700-1700nm.

"High" in hyperspectral images usually refers to high spectral resolution. For example, any point in a hyperspectral image collected through FS23 has 1200 spectral channels, which cover 400-1000nm, and each spectral channel corresponds to a different wavelength.

6.2 What is a frame rate (FrameRate)?

For FS-2X series line scan hyperspectral cameras, the frame rate refers to the number of lines acquired per second. For example, the maximum frame rate of the FS-23 is 128fps(frame per second), the spatial resolution is 1920, and the number of spectral channels is 1200,The spectral range is: 400-1000nm.

This means that the FS-23 can collect 128 lines per second, 1920 points per line, and 1200 spectral channels of data in the range of 400-1000nm per point.

6.3 What is the pixel format

The signal of each spectral channel at each point of the FS-2X series line scan hyperspectral camera is represented by a data, which can be an 8-bit binary number or a 16-bit binary number. When 16-bit binary numbers are selected, higher signal resolution is obtained, but larger data transmission bandwidth is occupied,The transmission speed slows down.

6.4 What is exposure time?

The exposure time refers to the time that the sensor receives the light signal. The longer the exposure time is, the larger the sampling result of the optical signal with the same intensity will be obtained. The smaller the

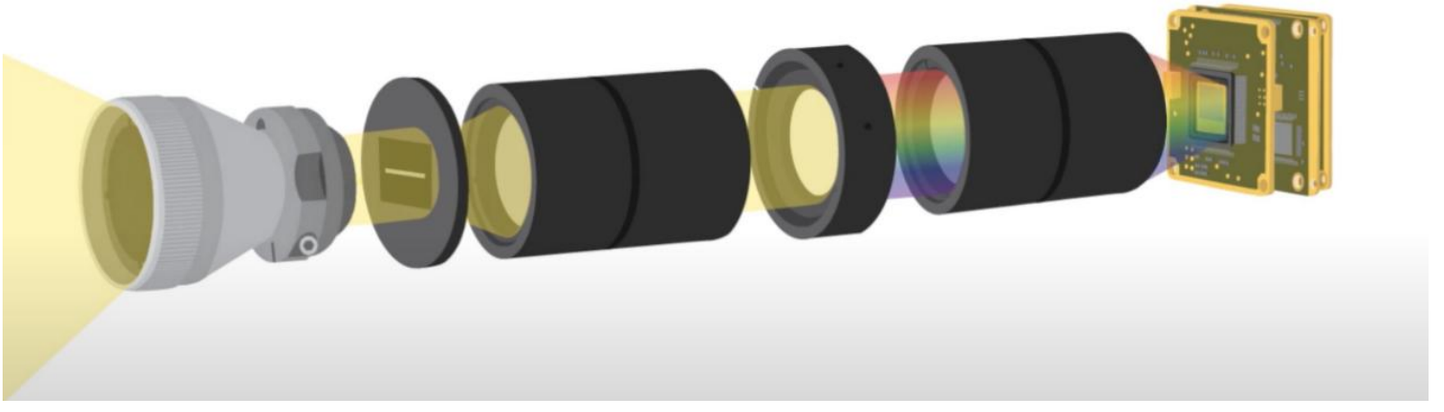
exposure time is, the smaller the sampling result of the optical signal with the same intensity will be obtained. When the sampling result is too small to effectively distinguish the information the user wants,It is suggested that the exposure time can be increased appropriately. When the sampling result is too large, resulting in the saturation of the signal (when the fluctuation of the optical signal cannot cause the change of the sampling signal), it is recommended to reduce the exposure time appropriately.

6.5 What is the number of spectral channels and the number of spatial pixels

Number of spectral channels: The number of spectral channels is the number of different wavelength ranges or bands recorded in a hyperspectral image. Hyperspectral imagery captures data on the reflection or radiation of an object over a continuous range of wavelengths. Each spectral channel represents a specific wavelength range, and different parts of the spectrum from visible to infrared can be included in the hyperspectral image.The more the number of spectral channels, the richer spectral information can be captured in the image, which is helpful to identify the characteristics of different ground features.

The number of spatial pixels: The number of spatial pixels refers to the resolution of the image in space, that is, the number of pixels in the image. Hyperspectral images are usually composed of pixels, each representing a specific small area on the surface of the Earth. The number of spatial pixels determines the detail and resolution of the image. Higher spatial resolution means that the image can show smaller features,Lower spatial resolution may not capture subtle features.

Together, the number of spectral channels and the number of spatial pixels determine the quality and information content of hyperspectral images. Hyperspectral images can provide rich spectral information and high-resolution spatial details, which can help in remote sensing applications for classification, environmental monitoring, agricultural management and other tasks.



7 Sample code

The following are some simple code examples that show how to use the SDK to control a hyperspectral camera to collect hyperspectral images and process hyperspectral image data.

7.1 Example 1 (the camera works in "line scan mode", the operation mode is ALL Mode, spectral dimension and spatial dimension 1 are combined to obtain 10 frames of hyperspectral data, and the hyperspectral data is converted into reflectivity data, and the data is saved as hyperspectral image files)

```
using Cameras;
using System;
using SystemCollectionsGeneric;
using SystemIO;
using SystemLinq;
using SystemText;
using SystemThreading;

namespace FS2XCamerasExample1
{
    /// <summary>
    /// 相机工作在"线扫描模式"下，The operation mode is ALL Mode, the spectral dimension and the spatial dimension 1 are combined to obtain 10 frames of hyperspectral data, and the hyperspectral data is converted into reflectivity data, and the data is saved as a hyperspectral image file.
/// The camera operates in "line scanning mode" and runs in ALL mode Spectral and spatial dimensions are merged to obtain 10 frames of hyperspectral data, which is then converted into reflectance data and saved as a hyperspectral image file
/// </summary>
internal class Program
{
static void Main(string[] args)
{
Execute();
ConsoleReadLine();
}
```

```
private async static void Execute()
{
List<ICameraInfo> cameraInfos;
try
{
// By calling the GetCameras method,您可以获取该电脑已连接的相机列表
                    //By calling the GetCameras method, you can obtain a list of cameras connected to the computer
                    cameraInfos = await FS2XEnumerateGetCameras();

                    ConsoleWriteLine($"Found {cameraInfosCount} cameras ");
                    for (int i = 0; i < cameraInfosCount; i++)
                    {
                        ConsoleWriteLine($"The {(i + 1)} camera is {cameraInfos[i]InstrumentSN}");
                    }
                }
                catch (Exception ex)
                {
                    ConsoleWriteLine(exMessage);
                    ConsoleReadLine();
                    return;
                }

                if (cameraInfosCount == 0)
                {
                    ConsoleWriteLine("Camera not found");
                    ConsoleReadLine();
                    return;
                }

                FS2XCamera camera;
                try
                {
                    //通过调用 InitCamera 方法，您可以创建指定相机的实例
                    //By calling the InitCamera method, you can create an instance of a specified camera
                    camera = await FS2XEnumerateInitCamera(cameraInfos[0], EnumWorkingMode
```

7.2 Example 2 (10 frames of hyperspectral data are obtained when the camera works in "line scan mode" and the operation mode is BANK mode)

```
Length - 1;

            //从"可选择的结束波长列表"中选择一个作为结束波长
            //Select one from the 'selectable end wavelength list' as the end wavelength
            float end = (float)infoBanklistCur[0]OptionalEndWavelengths[end_index]Value;

            //启用 BANK
            //Enable BANK
            infoBanklistCur[0]EnableBank(start, end);

            //停用 BANK
            //Disable BANK
            //infoBanklistCur[0]DisableBank()

            //设置工作模式为 1 个 BANK
            //Set the working mode to 1 bank
            cameraSetRunTypeBank(infoBanklistCur);
            ConsoleWriteLine("Set working mode");
        }

        //开始采集
        //Start collecting
        if (cameraIsOpen)
        {
            cameraStartGrab();
            //cameraStartGrab(infoFrameRateCur, (int)infoSpatialPixelNumCur, (int)infoBinSpatialPixelCur);
            ConsoleWriteLine("Start collecting");
        }

        while (grabDatasCount < 10)
        {
            ThreadSleep(10);
```

```
            }



            //停止采集
            //Stop collecting
            cameraStopGrab();
            ConsoleWriteLine("Stop collecting");

            //关闭相机
            //Turn off the camera
            cameraCloseCamera();
            ConsoleWriteLine("Turn off the camera");
        }
    }
}
```

7.3 Example 3 (the camera works in the "imaging hyperspectral mode", the operation mode is ALL Mode, the spectral dimension and the spatial dimension 1 are combined to obtain a hyperspectral image and save it as a hyperspectral image file)

```
using Cameras;
using System;
using SystemCollectionsGeneric;
using SystemIO;
using SystemText;
using SystemThreading;

namespace FS2XCamerasExample3
{
    /// <summary>
    ///
相机工作在"成像高光谱模式"下，运行模式为 ALL 模式,光谱维和空间维 1 合并，获得 1 幅高光谱图像，保存成高光谱图像
文件
```

```
    /// The camera operates in "Imaging Hyperspectral Mode" and runs in ALL mode The spectral dimension and spatial dimension are merged to obtain one hyperspectral image, which is saved as a hyperspectral image file
    /// </summary>
    internal class Program
    {
        static void Main(string[] args)
        {
            Execute();
            ConsoleReadLine();
        }
        private async static void Execute()
        {
            List<ICameraInfo> cameraInfos;
            try
            {
                //通过调用 GetCameras 方法，您可以获取该电脑已连接的相机列表
                //By calling the GetCameras method, you can obtain a list of cameras connected to the computer
                cameraInfos = await FS2XEnumerateGetCameras();

                ConsoleWriteLine($"Found {cameraInfosCount} cameras ");
                for (int i = 0; i < cameraInfosCount; i++)
                {
                    ConsoleWriteLine($"The {(i + 1)} camera is {cameraInfos[i]InstrumentSN}");
                }
            }
            catch (Exception ex)
            {
                ConsoleWriteLine(exMessage);
                ConsoleReadLine();
                return;
            }

            if (cameraInfosCount == 0)
            {
                ConsoleWriteLine("Camera not found");
                ConsoleReadLine();
                return;
            }
```

```
            FS2XCamera camera;
            try
            {
                //通过调用 InitCamera 方法，您可以创建指定相机的实例
                //By calling the InitCamera method, you can create an instance of a specified camera
                camera = await FS2XEnumerateInitCamera(cameraInfos[0],
EnumWorkingModeCAM_COMPUTER_CAM);
            }
            catch (Exception ex)
            {
                Console
```

7.4 Example 4 (1 Hyperspectral image is obtained when the camera is working in "imaging hyperspectral mode" and the operation mode is BANK mode)

```
using Cameras;
using System;
using SystemCollectionsGeneric;
using SystemThreading;

namespace FS2XCamerasExample4
{
    /// <summary>
    /// 相机工作在"成像高光谱模式"下，运行模式为 BANK 模式条件下，获得 1 幅高光谱图像
    /// The camera works in "imaging hyperspectral mode" and operates in BANK mode to obtain one
hyperspectral image
    /// </summary>
    internal class Program
    {

        static void Main(string[] args)
        {
            Execute();
            ConsoleReadLine();
        }
```

```
        private async static void Execute()
        {
            List<ICameraInfo> cameraInfos;
            try
            {
                //通过调用 GetCameras 方法，您可以获取该电脑已连接的相机列表
                //By calling the GetCameras method, you can obtain a list of cameras connected to the computer
                cameraInfos = await FS2XEnumerateGetCameras();

                ConsoleWriteLine($"Found {cameraInfosCount} cameras ");
                for (int i = 0; i < cameraInfosCount; i++)
                {
                    ConsoleWriteLine($"The {(i + 1)} camera is {cameraInfos[i]InstrumentSN}");
                }
            }
            catch (Exception ex)
            {
                ConsoleWriteLine(exMessage);
                ConsoleReadLine();
                return;
            }

            if (cameraInfosCount == 0)
            {
                ConsoleWriteLine("Camera not found");
                ConsoleReadLine();
                return;
            }

            FS2XCamera camera;
            try
            {
                //通过调用 InitCamera 方法，您可以创建指定相机的实例
                //By calling the InitCamera method, you can create an instance of a specified camera
                camera = await FS2XEnumerateInitCamera(cameraInfos[0],
EnumWorkingModeCAM_COMPUTER_CAM);
            }
```

```
            catch (Exception ex)
            {
                ConsoleWriteLine(exMessage);
                Console
```

7.5 Example 5 (Get 100 preview images and save to file)

```
BaseDirectory + "OutData₩₩";
            if (!DirectoryExists(directory))
            {
                DirectoryCreateDirectory(directory);
            }

            foreach (Bitmap bitmap in images)
            {
                bitmapSave(directory + DateTimeNowToString("yyyy-MM-dd HH:mm:ss fff") + "png",
ImageFormatPng);
            }
        }
    }
}
```

8 Camera Control Method (Camera Control Class)

8.1 Introducing Namespaces

Before you begin, make sure that you have installed the appropriate development environment and tools. And introduce all the namespaces of the camera to create camera instances and camera parameter instances.

```
using Cameras;
using FS2XCameras;
```

The Camera class "Camera" and the Camera Factory class "CameraFactory" are encapsulated in the namespaces of Cameras and Cameras.Factory, respectively. A Camera instance and a Camera parameter instance can be created only by introducing the current two namespaces.

Note: Developers need to copy the static library to their own program running directory. The location of the static library is: there are FigSpec-Cameras folders in the SDK directory, and all files in the Libraries Directory are static Libraries.

8.2 Get Camera List

By calling the GetCameras method, you can get a list of all available cameras currently connected to the PC.

```
try
{
        List<ICameraInfo> cameraInfos = await FS2XEnumerate.GetCameras();
}
catch (Exception ex)
{
    throw ex;
}
```

We recommend that you use the try catch keyword to capture the exception information generated during code execution and handle the exception information.

The return value CameraInfo class has the following parameters (note that the following parameters only represent the information of the built-in camera information, not the hyperspectral camera):

Parameter Name	Parameter Type	Parameter specification
DisplayModel	String type	Camera model
InstrumentSN	String type	Serial number of the internal camera
SoftwareVersion	Enumeration type	Software version number
HardwareVersion	Enumeration type	Hardware version number

8.3 Create a camera instance

By calling the InitCamera method we can get the Camera instance of the returned Camera, through which we operate on the Camera. Actions such as turning on the camera, turning off the camera, starting acquisition, stopping acquisition, receiving hyperspectral data, and receiving status events of the camera.

```
try
{
    camera = await FS2XEnumerate.InitCamera(cameraInfo,workingMode);
}
catch (Exception ex)
{
    throw ex;
}
```

We recommend that you use the try catch keyword to capture the exception information generated during code

execution and handle the exception information.

Camera information obtained by cameraInfo

Working mode of the camera

- 0: Image acquisition mode; The camera acquires a square plane
- 1: Line acquisition mode; The camera moves to the middle of the lens to acquire line hyperspectral data
- 2: IQ mode, note: This mode is unique to FS-IQ series cameras

8.4 Open Camera

We can open the camera by calling the OpenCamera method,

```
try
{
    camera.OpenCamera();
}
catch (Exception ex)
{
    throw ex;
}
```

We recommend that you use the try catch keyword to capture the exception information generated during code execution and handle the exception information.

8.5 Start image acquisition

Image acquisition was started using the StartGrab method. At this point, the camera will start capturing image data.

```
try
{
    camera.StartGrab();
    //camera.StartGrab(frameRate, num, binSpatialPixelCur);
}
catch (Exception ex)
{
    throw ex;
}
```

We recommend that you use the try catch keyword to capture the exception information generated during code execution and handle the exception information.

The collection method has 1 overloaded method:

The parameters of the overloaded method are as follows:

- frameRate: Floating-point; The frame rate of the current camera

- num: integer; The number of frames to be captured. It is recommended to use the space of the current camera.
- binSpatialPixelCur: integer; The number of spatial merges for the current camera

8.6 Receiving Hyperspectral Image Data

```
Camera. GrabedData + = (grabData) =>
{

};
```

The return value of the grabData class has the following parameters:

Parameter Name	Parameter Type	Parameter specification
Samples	Integer type	Number of space channels
Bands	Integer type	Number of spectral channels
PixelFormat	Enumeration type	The current format of hyperspectral image data. For example: Mono12
Values	List of 8-bit unsigned integral types	A frame of hyperspectral image data content.

Example of converting the returned Values to hyperspectral image data:

```
/// <summary>
/// Put the hyperspectral data into a three-dimensional array (datas [frame number] [spatial channel] [spectral channel])
/// Put hyperspectral data into a three-dimensional array (data [frame rate] [spatial channel] [spectral channel])
/// </summary>
/// <param name="grabDataList"></param>
private static ushort[,] SaveToArray(List<GrabData> grabDataList)
{
// 10 frames of hyperspectral data
//10 frames of hyperspectral data
int frameNum = grabDataList.Count;

      //建立一个三维数组 HSData[frameNum][samples][bands] 把数据放进去
      //Create a 3D array HSData [frameNum] [samples] [bands] and put the data in it
      ushort[,,] HSData = new ushort[frameNum, grabDataList[0].Samples, grabDataList[0].Bands];

      for (int f = 0; f < frameNum; f++)
      {
```

```
            GrabData grabData = grabDataList[f];
            for (int p = 0; p < grabData.Samples; p++)
            {
                for (int w = 0; w < grabData.Bands; w++)
                {
                    if (grabData.PixelFormat == EnumPixelFormat.Mono8)
                    {
                        HSData[f, p, w] = grabData.Values[p + w * grabData.Samples];
                    }
                    else
                    {
                        HSData[f, p, w] = BitConverter.ToUInt16(grabData.Values, (p + w * grabData.Samples) * 2);
}
}
}
}
Return HSData;
}
```

8.7 Stop image acquisition

Image acquisition was stopped using the StopGrab method.

```
try
{
    camera.StopGrab();
}
catch (Exception ex)
{
     throw ex;
}
```

We recommend that you use the try catch keyword to capture the exception information generated during code execution and handle the exception information.

8.8 Event handling

The Camera class provides four events, namely "Camera has been turned on event", "Camera has been turned off event", "Camera has started capturing event" and "Camera has stopped capturing event", which can be used to monitor Camera state changes.

```
camera.CameraOpened += () =>
{
    //相机已开启事件
    //Camera turned on event
};

camera.CameraClosed += () =>
{
    //相机已关闭事件
    //Camera turned off event
};

camera.GrabStarted += () =>
{
    //相机已开始采集事件
    //The camera has started capturing events
};

camera.GrabStopped += () =>
{
    //相机已停止采集事件
    //The camera has stopped collecting events
};
```

8.9 Turn off the camera

Use the CloseCamera method to close the camera.

```
try
{
    camera.CloseCamera();
}
catch (Exception ex)
{
    throw ex;
}
```

We recommend that you use the try catch keyword to capture the exception information generated during code execution and handle the exception information.

9 Get and set camera parameters

9.1 Obtain instrument parameter information

The camera.GetParmInfo method can be used to obtain all parameters of the current camera, including: camera model, serial number, frame rate, pixel format, exposure value, working mode, number of spatial channels, number of spectral channels, wavelength corresponding to the spectral channel, trigger mode and other parameters. Each time the user sets the parameters,It is recommended to read the actual parameters once, which has two functions:

1. Judge whether the parameter is consistent with the set value
2. Some parameters affect each other. For example, after setting the exposure time, the maximum frame rate will be affected, and the currently set frame rate may not be reached. At this time, the maximum frame rate will be automatically selected.

```
try
{
    ParmInfo   info = camera.GetParmInfo();
}
catch (Exception)
{
    throw;
}
```

We recommend that you use the try catch keyword to capture the exception information generated during code execution and handle the exception information.

Introduction to the fields of the ParmInfo class:

Parameter Name	Parameter Type	Parameter specification
FrameRateMin	Floating point type	Minimum supported frame rate, in fps(frame per second)
FrameRateMax	Floating point type	Maximum supported frame rate, in fps(frame per second)
FrameRateCur	Floating point type	Current frame rate, in fps(frame per second)
PixelFormatItems	Enumerated list	All supported pixel formats: ● Mono8 uses 8 bits per pixel ● Mono12 uses 12 bits per pixel ● Mono14 uses 14 bits per pixel The supported pixel formats returned by different cameras are different. Please refer to the returned pixel format list.

PixelFormatCur	Enumeration type	Current pixel format; E.g. Mono12
ExposureTimeMin	Floating point type	Minimum exposure value supported, in microseconds (us)
ExposureTimeMax	Floating point type	Maximum exposure value supported, in microseconds (us)
ExposureTimeCur	Floating point type	Current exposure value in microseconds (us)
RunType Items	Enumerated list type	The list of working modes supported by the currently used camera; (The working modes supported by different cameras are different, please refer to the list of working modes actually returned); ● A value of ALL indicates that ALL available spectral channels are used; ● When the value is SKIP, the SKIP mode is used. At this time, the number of spatial channels and the number of spectral channels will become 1/2 of the maximum value; ● When the value is BANK, it means that the BANK selection mode is used (the spectral range of interest can be selected for acquisition);
RunTypeCur	Enumeration type	Current working mode
BinSpatialSpecNotSame	Boolean	Does the camera support spatial merging and spectral merging with different parameters
BinSpatialPixelItems	Enumerated list type	The list of spatial pixel merging modes supported by the currently used camera (different cameras support different spatial pixel merging modes, please refer to the actual returned list) SpatialPixel1 represents space 1 cell merge SpatialPixel2 represents a spatial 2-cell merge SpatialPixel4 represents a spatial 4-cell merge SpatialPixel6 represents a spatial 6-cell merge. SpatialPixel8 represents a spatial 8-cell merge
BinSpatialPixelCur	Enumeration type	The spatial merge mode being used by the camera currently in use

BinSpectralItems	Enumerated list type	The list of spectral pixel merging modes supported by the camera currently in use (the spectral pixel merging modes supported by different cameras are different) SpectralChannel1 indicates spectral 1 pixel merging SpectralChannel2 indicates spectral 2-cell merging SpectralChannel4 indicates spectral 4-cell merging SpectralChannel6 indicates spectral 6-cell merging SpectralChannel8 indicates spectral 8-cell merging
BinSpectralCur	Enumeration type	Current Spectrum Merge Mode
Umcur spatial pixels	32-bit unsigned integer	Number of current spatial channels
Spatial pixels ummax	32-bit unsigned integer	Maximum number of spatial channels (without any merge)
Spectral channel numcur	32-bit unsigned integer	Total number of current spectral channels;
Spectral channel nummax	32-bit unsigned integer	Maximum total number of spectral channels in the camera; (without any merging)
SpectralChannelWavelength	List type	The wavelength corresponding to the spectral channel currently used by the camera. The number of elements in the list is the same as that of SpectralChannelNumCur.
SpectralChannelAllWavelength	List type	The wavelengths corresponding to all spectral channels supported by the camera currently in use. The number of elements in the list is consistent with that of SpectralChannelNumMax. From this list, you can know the maximum spectral range supported by the camera.
Banklistcur	Object Type List	The list of banks supported by the camera. (Different cameras support different number of banks, showing that several banks support several) The properties in the specific Bank are viewed in the method of setting the current running mode to BANK mode (SetRunTypeBank).
Trigger mode items	List type	All supported trigger modes, different cameras return different supported modes, please refer to the actual returned list: ● When the value is IN, it means internal trigger mode; ● When the value is OUT_RISING, it means the rising edge outer trigger mode; ● When the value is OUT_FALLING, it means the falling edge

		outer trigger mode;
TriggerModeCur	Enumeration type	Current trigger mode
GainMin	Floating point type	Minimum Gain Supported
GainMax	Floating point type	Maximum Gain Supported
GainCur	Floating point type	Current Gain
ReverseSpectral	Boolean	Current Inverse Spectrum Wavelength Status
ReverseSpatialPixel	Boolean	Current Reverse Space Pixel State

9.2 Get the camera working mode (GetWorkingMode)

mode is an enumeration type, which is described as follows:

- 0: Image acquisition mode; The camera acquires a square plane
- 1: Line acquisition mode; The camera moves to the middle of the lens to acquire line hyperspectral data
- 2: IQ mode, note: This mode is unique to FS-IQ series cameras

9.3 Setting the camera working mode (SetWorkingMode)

result is a Boolean type; true indicates that the operation succeeded, and false indicates that the operation failed.
mode is an enumeration type, which is described as follows:

- 0: Image acquisition mode; The camera acquires a square plane
- 1: Line acquisition mode; The camera moves to the middle of the lens to acquire line hyperspectral data
- 2: IQ mode, note: This mode is unique to FS-IQ series cameras

9.4 Set the camera to move to the preview position (SetCameraPreviewPosition)

Control the camera to move to the preview position

result is a Boolean type; true indicates that the operation succeeded, and false indicates that the operation failed.

9.5 Gets the automatic exposure time (GetAutoExposureValue)

result is a Boolean type; true indicates that the operation succeeded, and false indicates that the operation failed. signalRatio is a floating-point type; Range 0-1, default 0.5. Indicates the exposure time required when the signal amount reaches the set ratio, where the maximum exposure time during automatic exposure is 100000us

9.6 Set exposure time (SetExposureTime)

result is a Boolean type; true indicates that the operation succeeded, and false indicates that the operation failed. The exposureTime value is a floating point. The value of the setting must be greater than or equal to the minimum exposure value supported by the current camera and less than or equal to the maximum exposure value supported by the current camera. Exposure values are in microseconds (us).

9.7 Set Pixel Format (SetPixelFormat)

result is a Boolean type; true indicates that the operation succeeded, and false indicates that the operation failed. pixelFormat The supported pixel formats returned by different cameras are different, please select the pixel format list (ParmInfo.PixelFormatItems) in the actual returned pixel format list.

9.8 Set spectrum merge (SetBinSpectral)

Merge mode to reduce the spectral resolution to increase the amount of single pixel signal, and some models of hyperspectral cameras can increase the acquisition speed. Call the SetBinSpectral method to set the user's desired spectral pixel merge mode. Which allows setting items to be obtained through the BinSpectraItems method and view the supported merge modes

The binSpectral enumeration type field is introduced:

- SpectralChannel1 indicates spectral 1 pixel merging
- SpectralChannel2 indicates spectral 2-cell merging
- SpectralChannel4 indicates spectral 4-cell merging
- SpectralChannel6 indicates spectral 6-cell merging
- SpectralChannel8 indicates spectral 8-cell merging

Note: The spectral pixel merging options supported by different cameras are different. Please select them in the supported spectral merging List (ParmInfo.BinSpectralItems) actually returned. Some models of cameras only support the merging of spatial channels and spectral channels at the same time. It is recommended to reset the camera parameters after setting the parameters,View the actual merge method.

9.9 Set spatial merge (SetBinSpatialPixel)

result is a Boolean type; true indicates that the operation succeeded, and false indicates that the operation failed.

Merge modes to reduce spatial resolution to increase the amount of single-pixel signal, and some models of hyperspectral cameras can increase acquisition speed. Call the SetBinSpatialPixel method to set the user's desired spatial pixel merge mode. The items that are allowed to be set are obtained through the BinSpatialPixelItems method and the supported merge modes are viewed.

The binSpatialPixel enumeration type field is introduced:

SpatialPixel1 represents space 1 cell merge

SpatialPixel2 represents a spatial 2-cell merge

SpatialPixel4 represents a spatial 4-cell merge

SpatialPixel6 represents a spatial 6-cell merge.

SpatialPixel8 represents a spatial 8-cell merge

Note: The spatial pixel merge options supported by different cameras are different. Please select them in the supported spatial merge List (ParmInfo.BinSpatialPixelItems) actually returned. Some models of cameras only support the combination of spatial channels and spectral channels at the same time,It is recommended to reset the camera parameters after setting the parameters to view the actual merging method.

9.10 Set the current running mode to ALL Mode (SetRunTypeALL)

result is a Boolean type; true indicates that the operation succeeded, and false indicates that the operation failed.

Set the current operating mode to ALL Mode. When the operating mode is ALL, ALL available spectral channels and ALL spatial channels are used;

- ParmInfo.SpectralChannelNumCur can see the total number of spectral channels being used by the currently used camera;
- ParmInfo.SpectralChannelWavelength can see the wavelength corresponding to the spectral channel currently used by the camera. The number of elements in the list should be the same as that of ParmInfo.SpectralChannelNumCur.
- ParmInfo.SpatialPixelNumCur can see the number of spatial channels currently being used by the camera
- You can also view the frame rate speed supported in the current mode

9.11 Set the current running mode to SKIP mode (SetRunTypeSKIP)

result is a Boolean type; true indicates that the operation succeeded, and false indicates that the operation failed.

When the working mode is SKIP, it means to use all available spectral channels and all spatial channels;

9.12 Set the current running mode to BANK mode (SetRunTypeBank)

Important reminder: before setting the current working mode to BANK mode, be sure to obtain the available spectral channel, spatial channel and wavelength information corresponding to the spectral channel supported by the camera in the current state (the reason is: merging settings will cause changes in the wavelength information corresponding to the spectral channel, spatial channel and spectral channel);Set BANK mode according to this information

The zones parameter is a list of objects of type SpectralBankZone. Where the SpectralBankZone object contains the following properties:

Field Name	Type	Depaint
Index	Integer type	Read-only; Index values
Enable	Boolean	Read-only; Enabled
Wavelengths	Floating point list type	Read-only; Floating point lists
OptionalStartingWavelengths	Object List Type	Read-only; List of selectable starting wavelengths
OptionalEndWavelengths	Object List Type	Read-only; List of selectable end wavelengths
StartWavelength	Floating point type	Read-only; Starting wavelength
EndWavelength	Floating point type	Read only; Ending wavelengths
Offset	Object Class	Read-only; This class contains the maximum value (Max), minimum value (Min), minimum interval (Inc), and current offset (OffsetValue)
Size	Object Class	Read-only; Class contains maximum (Max), minimum (Min), minimum interval (Inc), and current size (SizeValue)
EnableBank	Method	Write-only; Execute this method to enable the Bank
DisableBank	Method	Write-only; Executing this method deactivates the Bank

9.13 Reverse space pixel

result is a Boolean type; true indicates that the operation succeeded, and false indicates that the operation failed. open is a Boolean type; true turns Spatial Pixel Inversion on, false turns Spatial Pixel Inversion off.

9.14 Set Gain

result is a Boolean type; true indicates that the operation succeeded, and false indicates that the operation failed. The gain value is a floating-point type. The value of the setting must be greater than or equal to the minimum gain value supported by the current camera and less than or equal to the maximum gain value supported by the current camera.

9.15 Get power (GetBattery)

battery is the object type, and the class field is introduced:

- State: charging State; Enumeration type 0 is normal, 1 is charging, 2 is discharging
- ElectricityLevel: remaining power ratio; Floating-point type;

The unit of frame rate is fps(frame per second).

9.16 Set the charging state (SetBattery)

result is a Boolean type; true indicates that the operation succeeded, and false indicates that the operation failed. discharge is a Boolean type; true turns on the backup battery to charge the MINI PC, false turns off the backup battery to charge the MINI PC