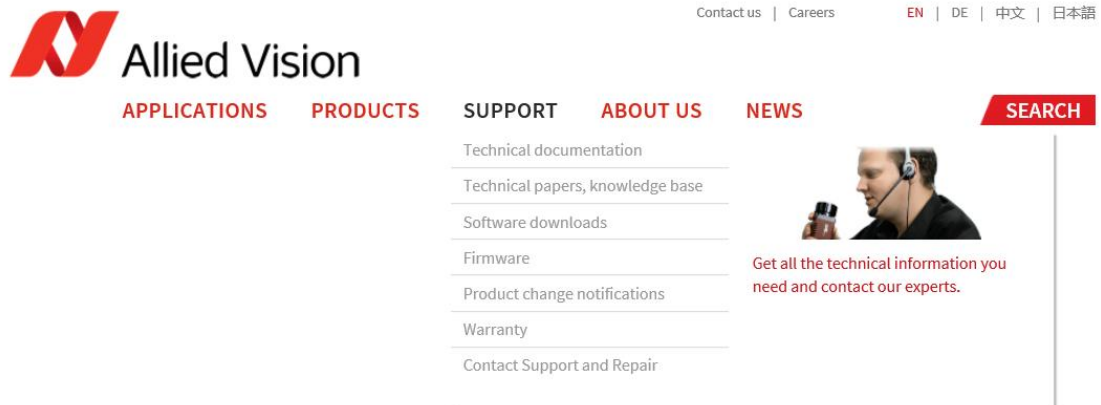


AVT 工业相机

——SDK 相关简介

第一部分：SDK 的下载和安装

1.1 请登录 AVT 官网：www.alliedvision.com 下载 SDK，免注册，免费



Choose Your Application Area



(图 1: www.alliedvision.com ->Support->Software downloads)

1.2 点击 [Vimba](#)，进入 Vimba 下载页面

SOFTWARE DOWNLOADS

Vimba SDK for GigE, IEEE1394, USB, and Camera Link cameras



[Vimba](#) is Allied Vision's future-proof and platform-independent SDK for GigE Vision, IEEE1394, USB3 Vision, and Camera Link cameras.

(图 2: Vimba 下载链接)

1.3 根据所用的操作系统，选择对应的 SDK 版本



Download Vimba

All Vimba downloads include programming examples and user manuals. You can download Vimba for free.

Downloads

Downloads for Windows:

[Vimba v2.1.3 Windows, Release Notes](#)

Downloads for Linux x86/x64:

[Vimba v2.1.3 Linux, Release Notes](#)

Downloads for ARMv7 32-bit:

[Vimba v2.1.3 ARM32, Release Notes](#)

Downloads for ARMv8 64-bit:

[Vimba v2.1.3 ARM64, Release Notes](#)

(图 3: 根据操作系统选择对应的下载版本)

Windows 版本的 SDK 支持以下操作系统:

Windows 7

Windows 8 and 8.1

Windows 10

1.4 以 Windows 为例，安装 SDK



（图 4：以 Windows 版本为例，安装界面如上，请选择 Application Development，表示完全安装）

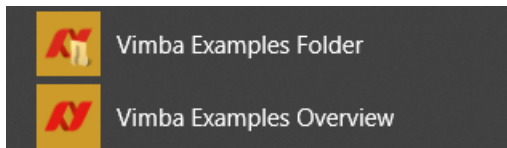
***注意：**早期的 Windows7 64bit 系统，因为数字签名认证的原因，需要先下载 Microsoft 的补丁 KB3033929 后再安装 SDK。

1.5 安装完成后，自动生成三个 Allied Vision 相关的文件夹

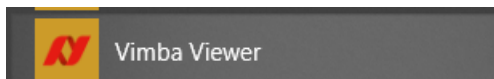


（图 5：安装 SDK 后，开始菜单中会出现三个文件夹）

1.6 其中常用的快速配置工具 VimbaViewer 和 SDK 相关的文档在 Allied Vision Vimba 文件夹中。



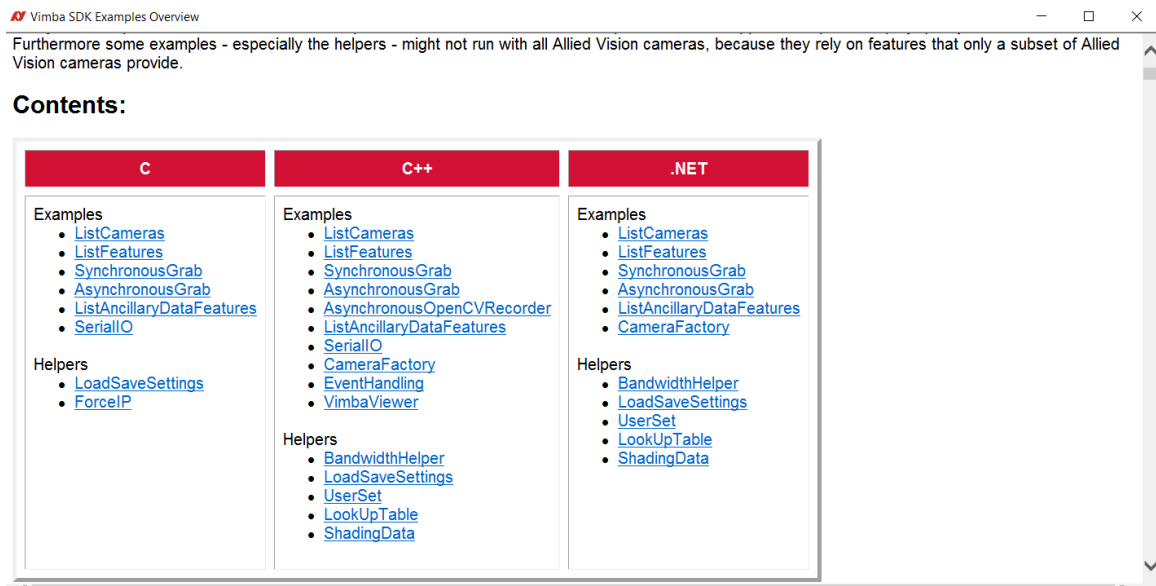
(图 6: 关于 Vimba 的例程相关)



(图 7: Vimba Viewer)







第二部分 Vimba SDK 开发介绍

2.1 进入 Allied Vision Vimba 文件夹中，点开 Vimba Examples Overview 可以看到按照不同的编程语言分类的例程，从最简单的遍历相机，遍历参数，到常用的同步采集，异步采集等例程，都有列出。直接点击某一个例程，会自动跳转到详细界面，点击界面会直接运行。



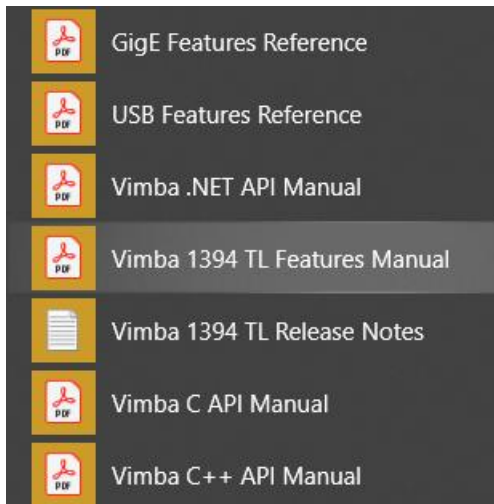
(图 8: Vimba Examples Overview)

2.2 如果要查看例程源代码，请在 Allied Vision Vimba 文件夹中，打开 Vimba Examples Folder

<input type="checkbox"/> Name	Date modified	Type	Size
 Images	9/5/2017 5:35 PM	File folder	
 VimbaC_Examples	9/5/2017 5:35 PM	File folder	
 VimbaCPP_Examples	9/5/2017 5:35 PM	File folder	
 VimbaCPP_Source	9/5/2017 5:35 PM	File folder	
 VimbaNET_Examples	9/5/2017 5:35 PM	File folder	
 ExamplesOverview	2/29/2016 3:42 PM	HTML Application	37 KB

(图 9: Vimba Examples Folder)

2.3 SDK 开发快速移植，请交叉参考例程源代码，以及对应的 API Manual 和 Gige Features Reference。例如开发语言为 C++，请打开 C++例程，同时参考《Vimba C++ API Manual》、《Gige Features Reference》



(图 10: Allied Vision Vimba 文件夹内的说明文档)

2.4 SDK 基本框架（以 C 语言为例）

1. 打开 Vimba

```
VmbStartup();
```

2. 遍历相机

```
bool bGigE;
VmbUInt32_t nCount;
VmbCameraInfo_t* pCameras;

// We ask Vimba for the presence of a GigE transport layer
VmbError_t err = VmbFeatureBoolGet( gVimbaHandle, "GeVTLLIsPresent", &bGigE );
if ( VmbErrorSuccess == err )
{
    if ( true == bGigE )
    {
        // We use all network interfaces using the global Vimba handle
        err = VmbFeatureCommandRun( gVimbaHandle, "GeVDiscoveryAllOnce" );
    }
}
if ( VmbErrorSuccess == err )
{
    // Get the number of connected cameras
    err = VmbCamerasList( NULL, 0, &nCount, sizeof *pCameras );

    if ( VmbErrorSuccess == err )
    {
        // Allocate accordingly
        pCameras = (VmbCameraInfo_t*)malloc( nCount * sizeof *pCameras );
        // Get the cameras
        err = VmbCamerasList( pCameras, nCount, &nCount, sizeof *pCameras );
        // Print out each camera's name
        for ( VmbUInt32_t i=0; i<nCount; ++i )
        {
            printf( " %s\n", pCameras[i].cameraName );
        }
    }
}
```

3. 打开相机

```
VmbCameraInfo_t *pCameras;
VmbHandle_t hCamera;

// Get all known cameras as described in chapter "Listing available cameras"

// Open the first camera
if ( VmbErrorSuccess == VmbCameraOpen( pCameras[0].cameraIdString,
                                       VmbAccessModeFull, hCamera ) )
{
    printf( "Camera opened, handle [0x%p] retrieved.\n", hCamera );
}
```

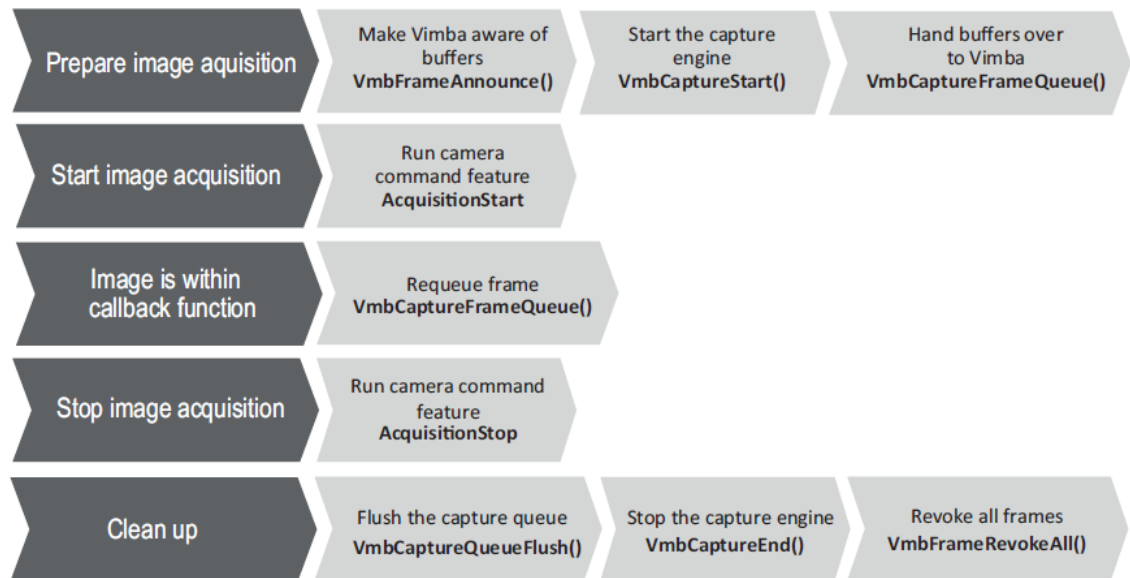

4. 读取/设置参数

Feature Type	Operation	Function
Enumeration	Set	VmbFeatureEnumSet
	Get	VmbFeatureEnumGet
	Range	VmbFeatureEnumRangeQuery
	Other	VmbFeatureEnumIsAvailable
		VmbFeatureEnumAsInt
VmbFeatureEnumAsString		
VmbFeatureEnumEntryGet		
Integer	Set	VmbFeatureIntSet
	Get	VmbFeatureIntGet
	Range	VmbFeatureIntRangeQuery
	Other	VmbFeatureIntIncrementQuery
Float	Set	VmbFeatureFloatSet
	Get	VmbFeatureFloatGet
String	Set	VmbFeatureStringSet
	Get	VmbFeatureStringGet
	Range	VmbFeatureStringMaxlengthQuery
Boolean	Set	VmbFeatureBoolSet
	Get	VmbFeatureBoolGet
Command	Set	VmbFeatureCommandRun
	Get	VmbFeatureCommandIsDone
Raw data	Set	VmbFeatureRawSet
	Get	VmbFeatureRawGet

5. 图像采集

Image capture and image acquisition are two independent operations: **Vimba API captures** images, the **camera acquires** images.

To obtain an image from your camera, setup Vimba API to capture images before starting the acquisition on the camera:



2.5 客户问题回复

1. SDK 中如何设置相机工作在软件触发/硬件触发模式？

(以下省略…打开 Vimba, 遍历相机, 打开相机的过程…)

软件触发模式:

设置 AcquisitionMode 为 Continuous:

```
VmbFeatureEnumSet(VmbHandle_t, "AcquisitionMode", "Continuous");
```

设置 TriggerMode 为 On:

```
VmbFeatureEnumSet(VmbHandle_t, "TriggerMode", "On");
```

设置 TriggerSelector 为 FrameStart:

```
VmbFeatureEnumSet(VmbHandle_t, "TriggerSelector", "FrameStart");
```

设置 TriggerSource 为 Software:

```
VmbFeatureEnumSet(VmbHandle_t, "TriggerSource", "Software");
```

开始采集:

```
VmbFeatureCommandRun(VmbHandle_t, "AcquisitionStart");
```

给触发信号:

```
VmbFeatureCommandRun(VmbHandle_t, "TriggerSoftware");
```

完成

外部触发模式:

设置 AcquisitionMode 为 Continuous:

```
VmbFeatureEnumSet(VmbHandle_t, "AcquisitionMode", "Continuous");
```

设置 TriggerMode 为 On:

```
VmbFeatureEnumSet(VmbHandle_t, "TriggerMode", "On");
```

设置 TriggerSelector 为 FrameStart:

```
VmbFeatureEnumSet(VmbHandle_t, "TriggerSelector", "FrameStart");
```

设置 TriggerSource 为 Line1:

```
VmbFeatureEnumSet(VmbHandle_t, "TriggerSource", "Line1");
```

开始采集:

```
VmbFeatureCommandRun(VmbHandle_t, "AcquisitionStart");
```

完成

软件触发和外部触发切换时, 需要先停止采集, 然后设置相应的参数, 然后再开始采集;

2. SDK 中如何读取和设置相机曝光时间, 增益?

设置 ExposureMode 为 Timed:

```
VmbFeatureEnumSet(VmbHandle_t, "ExposureMode", "Timed");
```

读取曝光时间:

```
VmbFeatureFloatGet(VmbHandle_t, "ExposureTimeAbs", &exposure_time);
```

设置曝光时间:

```
VmbFeatureFloatSet(VmbHandle_t, "ExposureTimeAbs", exposure_time);
```

读取增益:

```
VmbFeatureFloatGet(VmbHandle_t, "Gain", &gain_value);
```

设置增益:

```
VmbFeatureFloatSet(VmbHandle_t, "Gain", gain_value);
```

3. ROI 设置?

设置图像高度:

```
VmbFeatureIntSet(VmbHandle_t, "Height", height_value);
```

设置图像宽度:

```
VmbFeatureIntSet(VmbHandle_t, "Width", width_value);
```

设置图像 X offset:

```
VmbFeatureIntSet(VmbHandle_t, "OffsetX", xoffset_value);
```

设置图像 Y offset:

```
VmbFeatureIntSet(VmbHandle_t, "OffsetY", yoffset_value);
```

4. 设置图像镜像？

水平镜像：

```
VmbFeatureBoolSet(VmbHandle_t, "ReverseX", true);
```

垂直镜像：

```
VmbFeatureBoolSet(VmbHandle_t, "ReverseY", true);
```

5. 黑白/彩色相机格式切换？

AVT 相机支持以下图像格式，通过修改 PixelFormat 的值来修改图像格式，注意：不是所有的型号都支持以下全部格式，请参考对应型号的技术手册。

PixelFormat

Origin of feature	Camera
Type	Enumeration
Access	R/W
Possible values	<i>Mono8, Mono10, Mono12, Mono12Packed, Mono14, BayerGB8, BayerRG8, BayerGR8, BayerBG8, BayerBG10, BayerGB12Packed, BayerGR12Packed, BayerGB12, BayerRG12, BayerGR12, RGB8Packed, BGR8Packed, RGBA8Packed, BGRA8Packed, RGB12Packed, YUV411Packed, YUV422Packed, YUV444Packed</i>
Affected features	BinningHorizontal, StreamHoldCapacity, PayloadSize, NonImagePayloadSize, WidthMax, ImageSize, AcquisitionFrameRateAbs, ExposureTimeAbs, AcquisitionFrameRateLimit, Width, OffsetX, BinningVertical, HeightMax, Height, OffsetY

6. 如何判断当前相机是黑白还是彩色？

通过读取相机的型号，判断当前相机是黑白还是彩色，彩色相机的型号中带有字母“C”

```
VmbFeatureStringGet(VmbHandle_t, "DeviceModelName", &name_string);
```

通过判断 name_string 中是否有字母“C”来判断

DeviceModelName

Origin of feature	Camera
Type	String
Access	R/C
Affected features	N/A

Camera model name. Software should use the DevicePartNumber to distinguish between models.

Example: GT2450C

7. 回调函数中如何获得图像数据:

Vimba 中典型的回调函数写法:

```
// The callback that gets executed on every filled frame
void VMB_CALL FrameDoneCallback( const VmbHandle_t hCamera, VmbFrame_t *pFrame )
{
    if ( VmbFrameStatusComplete == pFrame->receiveStatus )
    {
        printf( "Frame successfully received\n" );
    }
    else
    {
        printf( "Error receiving frame\n" );
    }
    VmbCaptureFrameQueue( hCamera, pFrame, FrameDoneCallback );
}
```

通过判断 pFrame 结构体中的 receiveStatus 来确定当前帧是否完整，如果完整则进入取图像数据的部分代码，传入回调函数的结构体参数 VmbFrame_t，包含以下成员，通过成员名可以直接获得当前帧的各类信息，其中，图像数据头地址是 void* buffer

```
typedef struct
{
    //----- In -----
    void*          buffer;
    VmbUint32_t    bufferSize;

    void*          context[4];

    //----- Out -----
    VmbFrameStatus_t    receiveStatus;
    VmbFrameFlags_t     receiveFlags;

    VmbUint32_t    imageSize;
    VmbUint32_t    ancillarySize;

    VmbPixelFormat_t    pixelFormat;

    VmbUint32_t    width;
    VmbUint32_t    height;
    VmbUint32_t    offsetX;
    VmbUint32_t    offsetY;

    VmbUint64_t    frameID;
    VmbUint64_t    timestamp;
} VmbFrame_t;
```

