

Java 接口开发说明书



本手册中所提及的其它软硬件产品的商标与名称，都属于相应公司所有。

本手册的版权属于中国大恒（集团）有限公司北京图像视觉技术分公司所有。未得到本公司的正式许可，任何组织或个人均不得以任何手段和形式对本手册内容进行复制或传播。

本手册的内容若有任何修改，恕不另行通知。

© 2019 中国大恒（集团）有限公司北京图像视觉技术分公司版权所有

网 站：<http://www.daheng-imaging.com>

销售信箱：sales@daheng-imaging.com

销售热线：010-82828878 转 8068

支持信箱：support@daheng-imaging.com

支持热线：400-999-7595

目 录

1	相机工作流程.....	1
1.1	整体工作流程	1
1.1.1	普通枚举方式.....	1
1.1.2	无 Root 权限枚举	2
1.1.3	枚举方式的选择	3
1.2	图像采集流程	3
1.2.1	getRawImage 方法	3
1.2.2	getBitmap 方法	6
1.2.3	getImageBySurface 方法	7
1.3	功能控制流程	8
1.3.1	字符串控制方式	8
1.3.2	属性访问方式 (旧接口)	8
1.4	整体示例代码	9
1.4.1	普通枚举方式.....	9
1.4.2	无 Root 权限枚举.....	10
2	环境搭建.....	12
2.1	Android	12
2.1.1	工程创建	12
2.1.2	库的导入	14
2.1.3	javadoc 的配置	16
3	编程指引.....	18
3.1	引入库.....	18
3.2	枚举设备	18
3.2.1	普通方式.....	19
3.2.2	无 Root 权限方式.....	19
3.3	打开设备	20
3.4	采集控制	21
3.5	图像采集与处理	22
3.5.1	getRawImage 方法.....	22
3.5.2	getBitmap 方法	28

3.5.3	getImageBySurface 方法	30
3.5.4	图像质量提升	33
3.6	相机控制	33
3.6.1	属性节点字符串模式访问	33
3.6.2	属性直接访问（旧接口）	36
3.7	导入导出相机配置	39
3.8	异常处理	40
3.8.1	示例代码	40
3.8.2	异常类型	41
4	常见问题解答	42
5	附录	43
5.1	属性参数	43
5.1.1	设备属性参数	43
5.1.2	流属性参数	50
5.2	功能类定义	51
5.2.1	Feature	51
5.2.2	IntFeature	53
5.2.3	FloatFeature	55
5.2.4	EnumFeature	58
5.2.5	BoolFeature	60
5.2.6	StringFeature	61
5.2.7	BufferFeature	62
5.2.8	CommandFeature	64
5.2.9	StaticDefectCorrection	64
	图像数据格式，见 PixelColorFilterEntry	65
5.3	数据类型定义	66
5.3.1	DeviceClass	66
5.3.2	AccessStatus	66
5.3.3	AccessMode	66
5.3.4	FrameStatus	66
5.3.5	PixelFormatEntry	66
5.3.6	PixelSizeEntry	67
5.3.7	PixelColorFilterEntry	67
5.3.8	AcquisitionModeEntry	68
5.3.9	TriggerSourceEntry	68
5.3.10	TriggerActivationEntry	68
5.3.11	ExposureModeEntry	68

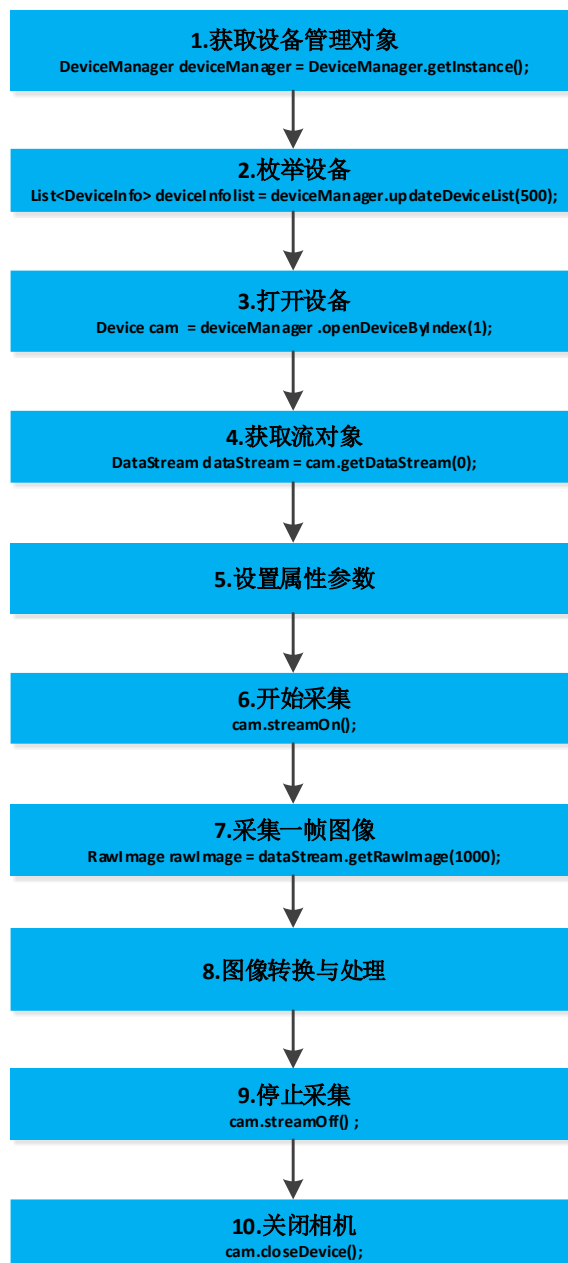
5.3.12	UserOutputSelectorEntry	68
5.3.13	UserOutputModeEntry	68
5.3.14	GainSelectorEntry	69
5.3.15	BlackLevelSelectEntry	69
5.3.16	BalanceRatioSelectorEntry	69
5.3.17	AALightEnvironmentEntry	69
5.3.18	UserSetEntry	69
5.3.19	AWBLampHouseEntry	70
5.3.20	TestPatternEntry	70
5.3.21	TriggerSelectorEntry	70
5.3.22	LineSelectorEntry	70
5.3.23	LineModeEntry	71
5.3.24	LineSourceEntry	71
5.3.25	LutSelectorEntry	71
5.3.26	TransferControlModeEntry	71
5.3.27	TransferOperationModeEntry	71
5.3.28	TestPatternGeneratorSelectorEntry	71
5.3.29	ChunkSelectorEntry	71
5.3.30	BinningHorizontalModeEntry	72
5.3.31	BinningVerticalModeEntry	72
5.3.32	SensorShutterModeEntry	72
5.3.33	AcquisitionStatusSelectorEntry	72
5.3.34	GammaModeEntry	72
5.3.35	ColorTransformationModeEntry	72
5.3.36	ColorTransformationValueSelectorEntry	73
5.3.37	AutoEntry	73
5.3.38	SwitchEntry	73
5.3.39	RegionSendModeEntry	73
5.3.40	RegionSelectorEntry	73
5.3.41	BayerConvertType	74
5.3.42	ValidBit	74
5.3.43	ImageMirrorMode	74
5.3.44	ActualBits	74
5.3.45	TimerSelectorEntry	74
5.3.46	TimerTriggerSourceEntry	75
5.3.47	CounterSelectorEntry	75
5.3.48	CounterEventSourceEntry	75
5.3.49	CounterResetSourceEntry	75
5.3.50	CounterResetActivationEntry	75
5.4	接口定义	75
5.4.1	DeviceManager	75
5.4.2	Device	80
5.4.3	DataStream	83
5.4.4	RawImage	86

5.4.5	RGBImage.....	90
5.4.6	ARGBImage	93
5.4.7	ByteBuffer.....	96
5.4.8	Utility.....	98
5.4.9	ImageFormatConvert	99
5.4.10	ImageProcess	105
6	版本说明.....	110

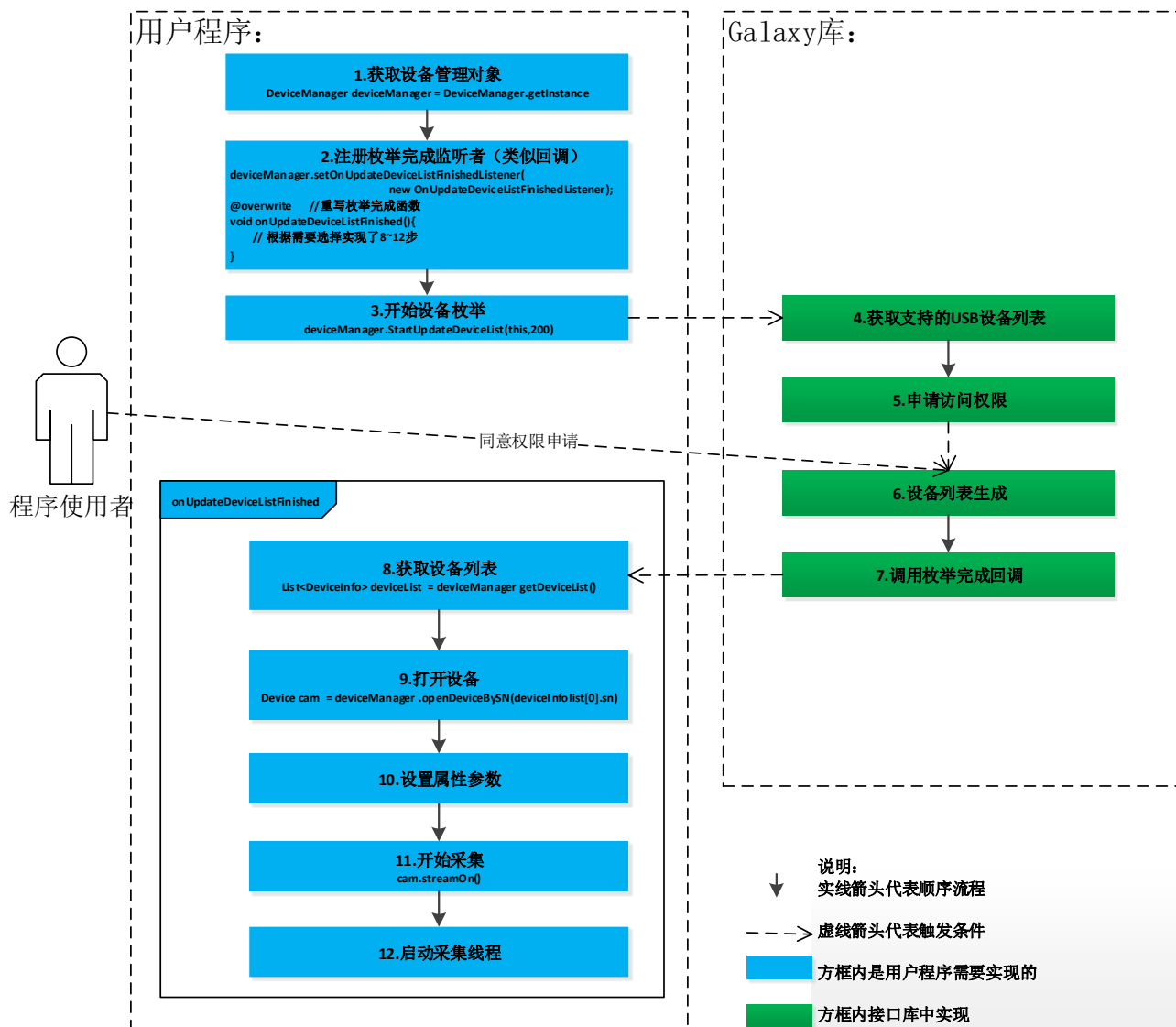
1 相机工作流程

1.1 整体工作流程

1.1.1 普通枚举方式



1.1.2 无 Root 权限枚举



说明:

- 1) 图中第 8~12 步, 演示了枚举完成后, 在重写 (override) 的枚举完成响应函数中可以进行的操作, 但不局限于这些操作;
- 2) **当 Android 系统未连接 U3V 相机时, 不能使用这种枚举方式;**
- 3) **不支持在多个线程中同时开始无 Root 权限枚举的过程。**

1.1.3 枚举方式的选择

1.1.3.1.1 非 Android 系统

使用普通枚举方式。

1.1.3.1.2 Android 系统

- 1) 在默认的 Android 系统下, 普通应用程序没有访问 USB 设备的权限, 使用**无 Root 权限枚举**方式;
- 2) 在定制的 Android 系统下, 应用程序拥有了访问 USB 设备的权限, 使用**普通枚举**方式;
- 3) 当用户只使用非 U3V 相机时, 使用**普通枚举**方式。

1.2 图像采集流程

1.2.1 getRawImage 方法

1.2.1.1 可选流程一



说明:

1. 流程中的 2~7 步是可选流程, 根据需求在程序中进行选择实现;
2. 可以满足的图像处理需求:
 - 1) 可以对输出原始图像数据进行处理和保存;
 - 2) 可以对 RGB 数据进行颜色校正、Gamma、对比度等处理;

- 3) 可以根据需要对 RGB 数据进行其他的处理，每一个像素使用 3 个 Byte 保存，顺序是 R->G->B;
(对应第 5 步)
- 4) 可以得到 Bitmap 格式的数据，并进行需要的处理和显示。

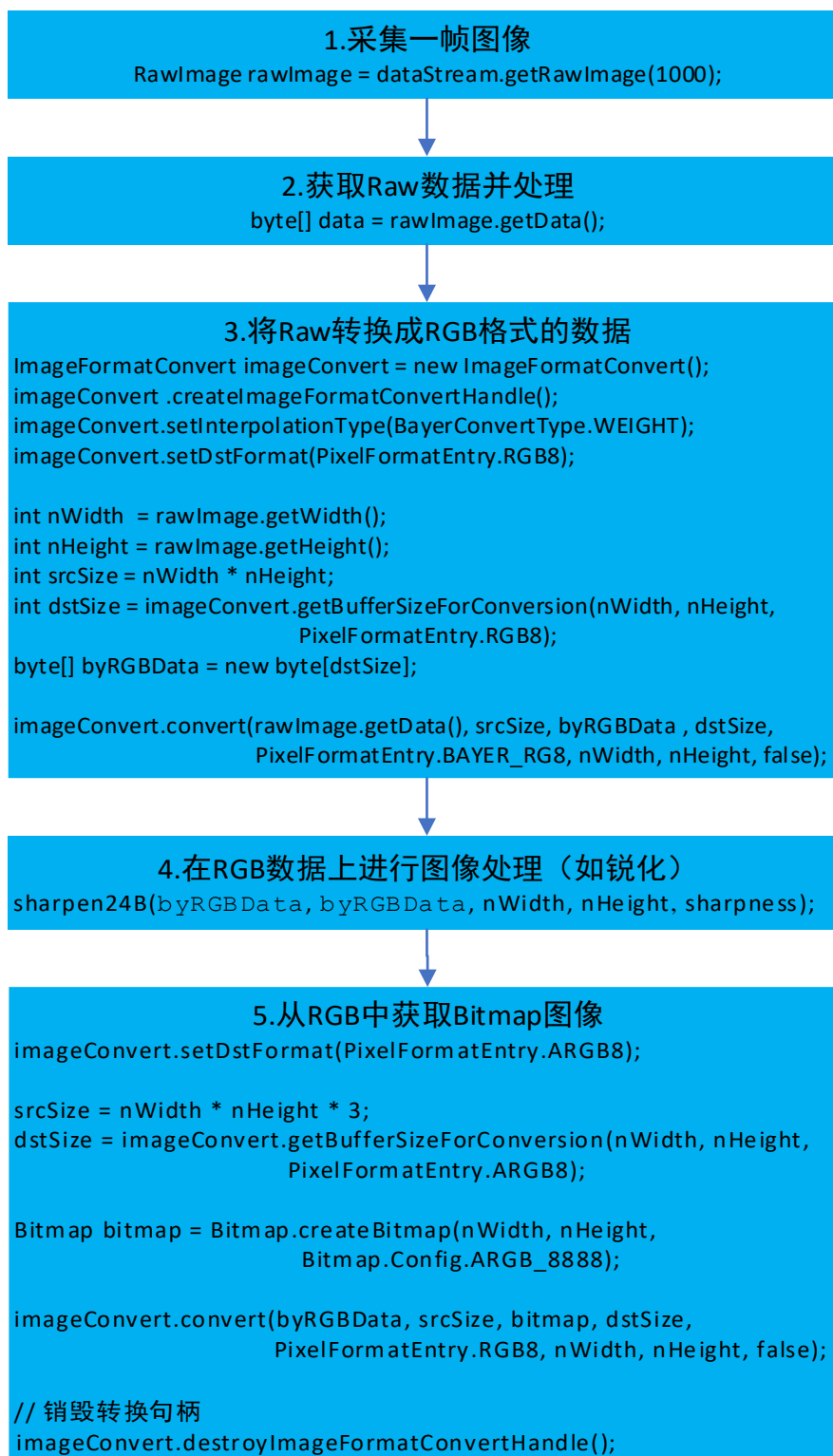
1.2.1.2 可选流程二



说明：

1. 流程中的 2~7 步是可选流程，根据需求在程序中进行选择实现；
2. 可以满足的图像处理需求：
 - 1) 可以对输出原始图像数据进行处理和保存；
 - 2) 可以进行颜色校正、Gamma、对比度等处理；（对应第 4 步）
 - 3) 可以得到 Bitmap 格式的数据，并进行需要的处理和显示。

1.2.1.3 可选流程三

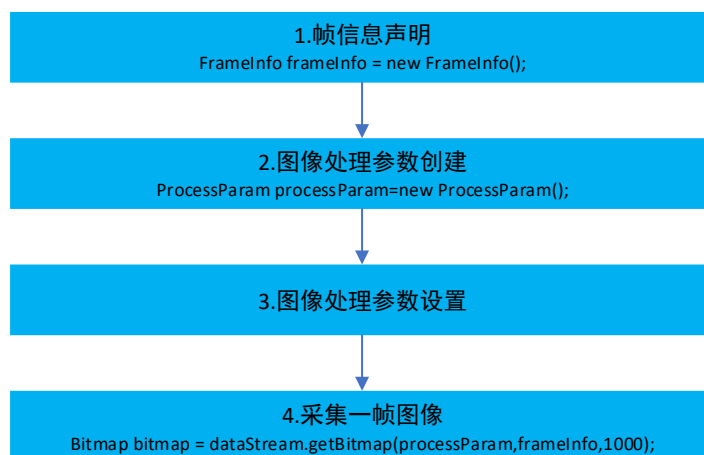


说明：

1. 流程中的第 4 步是可选流程，根据需求在程序中进行选择实现；
2. 可以满足的图像处理需求：
 - 1) 可以对输出原始图像数据进行处理和保存；
 - 2) 可以进行锐化等处理；（对应第 4 步）

- 3) 可以得到 Bitmap 格式的数据，并进行需要的处理和显示。

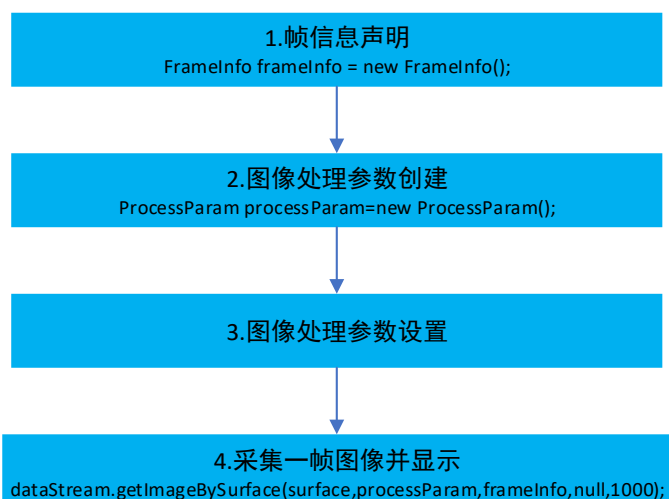
1.2.2 getBitmap 方法



说明：

1. 流程中的 1~2 步，建议在开始采集时，进行一次即可；
2. 步骤 3 可选，如果不进行则使用默认的图像采集与处理和参数；**注意：默认不进行颜色校正、Gamma、对比度等处理；**
3. 可以满足的图像处理需求：
 - 1) 可以进行颜色校正、Gamma、对比度等处理；
 - 2) 可以得到 Bitmap 格式的数据，并进行需要的处理和显示；
4. 优点：
 - 1) 零拷贝采集图像效率高；
 - 2) 调用简单。

1.2.3 getImageBySurface 方法

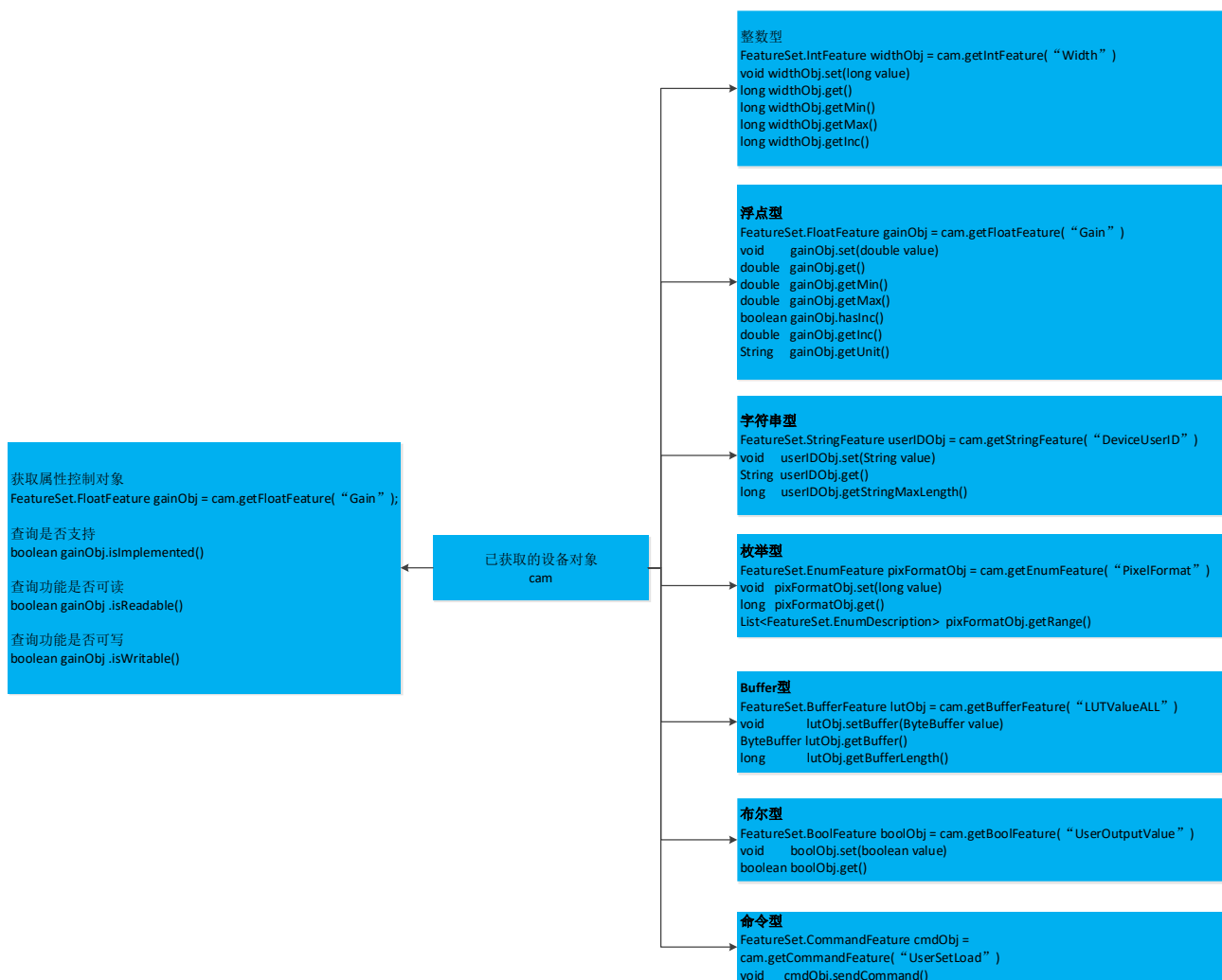


说明：

1. 流程中的 1~2 步，建议在开始采集时，进行一次即可；
2. 步骤 3 可选，如果不进行则使用默认的图像采集与处理和参数；**注意：默认不进行颜色校正、Gamma、对比度等处理；**
3. 可以满足的图像处理需求：
 - 1) 可以进行颜色校正、Gamma、对比度等处理；
 - 2) 将显示在指定界面控件上；
4. 优点：
 - 1) 零拷贝采集图像效率高；
 - 2) 调用简单。

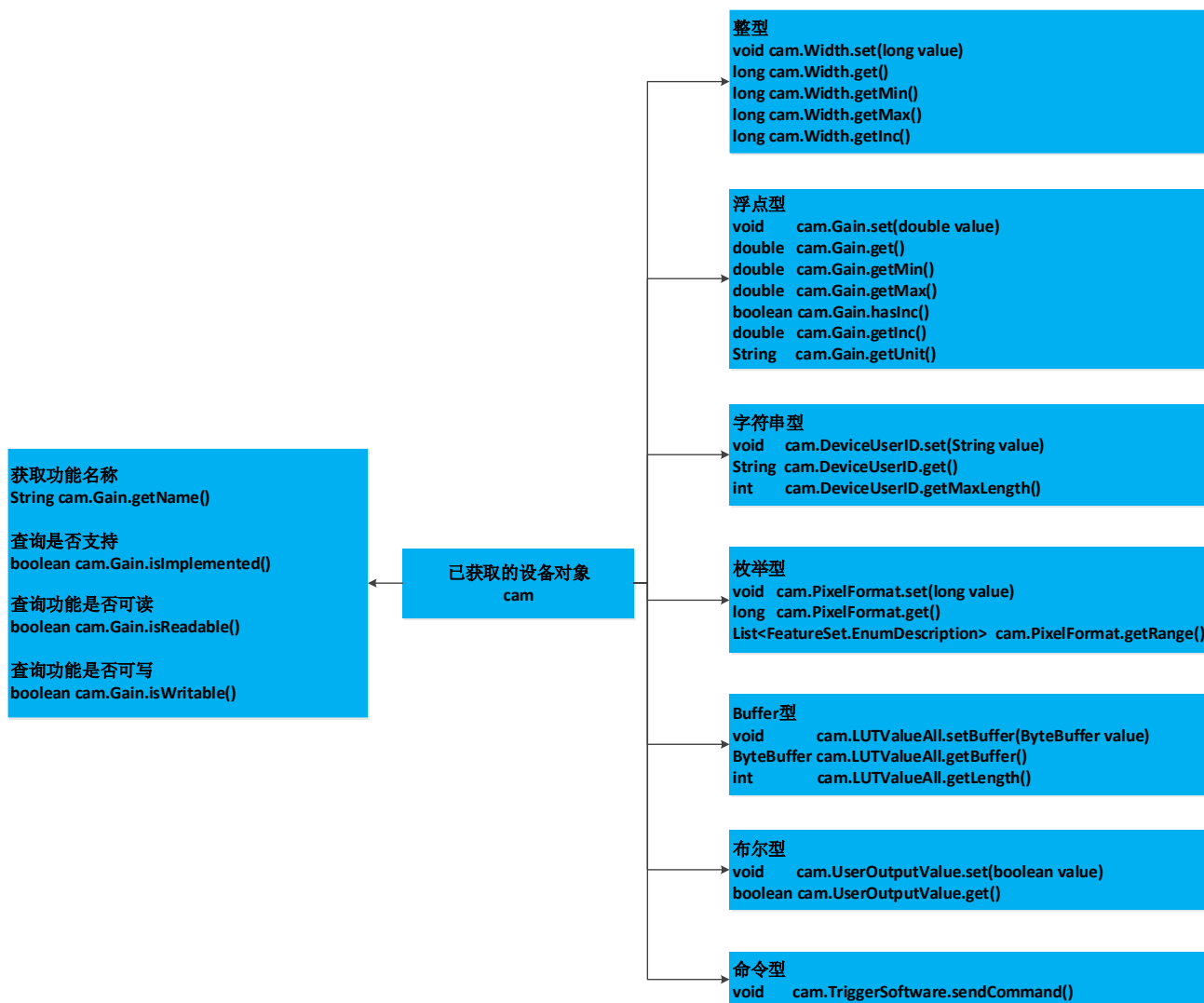
1.3 功能控制流程

1.3.1 字符串控制方式



1.3.2 属性访问方式（旧接口）

Java 旧接口方式访问相机，对于一些新的功能节点，如 Sensor 宽高信息、环境光源预设、黑电平、PGA 增益、用户数据区操作、2D 降噪、多源触发、序列、平场矫正功能、Burst 功能、串口操作等，旧接口不在升级和维护，遇到这些属性设置时候，请使用[字符串模式访问](#)。



1.4 整体示例代码

1.4.1 普通枚举方式

```
// 导入设备库
import galaxy.Device;
import galaxy.DeviceManager;

// 获取设备管理实例
DeviceManager deviceManager= DeviceManager.getInstance();

// 枚举设备，参数为超时时间，返回枚举到的设备信息列表
List<DeviceInfo> devInfoList = deviceManager.updateDeviceList(200);
if (devInfoList.size() == 0)
{
    // 提示没有枚举到设备
}
```

```

    return;
}

// 打开第一个设备
// 通过索引打开设备，可以传入 1、2、3.....，需要注意索引从 1 开始
Device cam = deviceManager.openDeviceByIndex(1);

// 获取第 0 个流通道对象，当前支持的相机只有一个流通道
DataStream dataStream = cam.getDataStream(0);

// 根据需要设置参数 .....

// 开始采集
cam.streamOn();

// 循环进行图像数据采集和处理
for(int i = 0; i < 10; ++i){
    RawImage rawImage = null;
    try{
        // 获取一幅图像，参数是超时时间，单位是 ms
        rawImage = dataStream.getRawImage(500);
    }
    catch(ExceptionSet.Timeout timeout) {
        // 捕获到超时，直接继续
        continue;
    }
    catch(ExceptionSet exceptionSet) {
        // 捕获到其他异常，打印异常信息后，继续尝试获取图像或者退出循环
        exceptionSet.printStackTrace();
        // continue;
    }
    // 图像的处理.....
}

// 停止采集
cam.streamOff();

// 关闭设备
cam.closeDevice();

```

1.4.2 无 Root 权限枚举

```

// 导入设备库
import galaxy.Device;
import galaxy.DeviceManager;

```



```
// 获取设备管理实例
DeviceManager deviceManager= DeviceManager.getInstance();

// 重写枚举完成函数
deviceManager.setOnUpdateDeviceListFinishedListener(new
DeviceManager.OnUpdateDeviceListFinishedListener() {
    @Override
    public void onUpdateDeviceListFinished(){
        List<DeviceInfo> deviceInfoList = deviceManager.getDeviceInfoList();
        if(deviceInfoList.size() == 0){
            // 提示未枚举到设备（或无设备接入）
        }
        else{
            // 进行打开、开采等操作，代码同“普通枚举方式”中的打开、开采等操作
        }
    }
});

// 开始枚举，只有在 MainActivity 中第一个参数 context 才能传入 this
deviceManager.startUpdateDeviceList(this, 200);
```

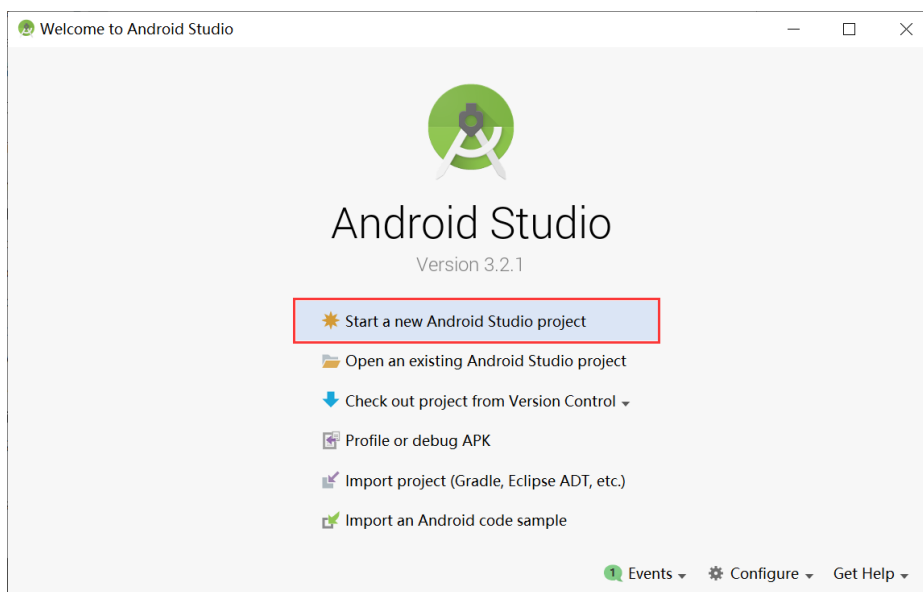
2 环境搭建

2.1 Android

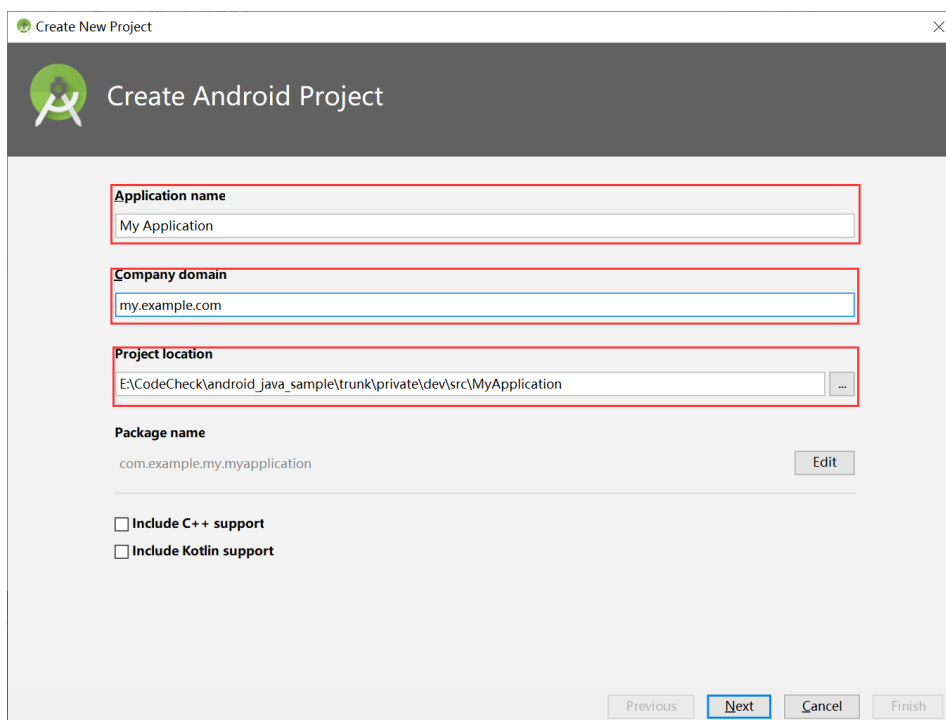
本文档以 Android Studio 3.2.1 版本为例进行开发环境搭建过程的介绍。

2.1.1 工程创建

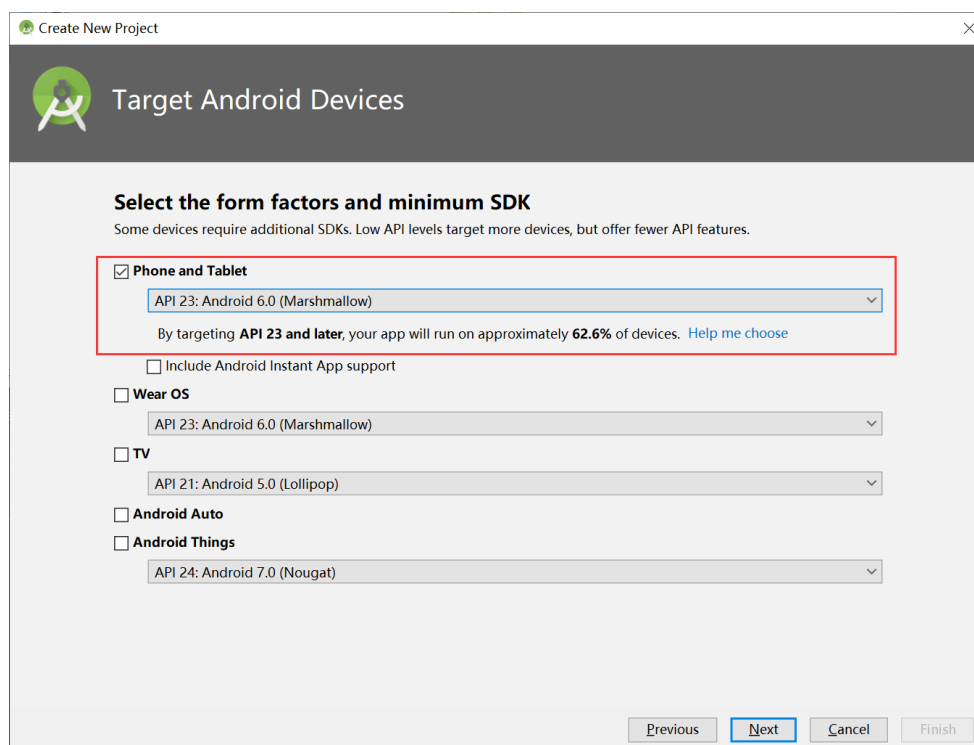
- 1) 在 AdnroidStudio 新建工程页中选择 “start a new Android Studio project”



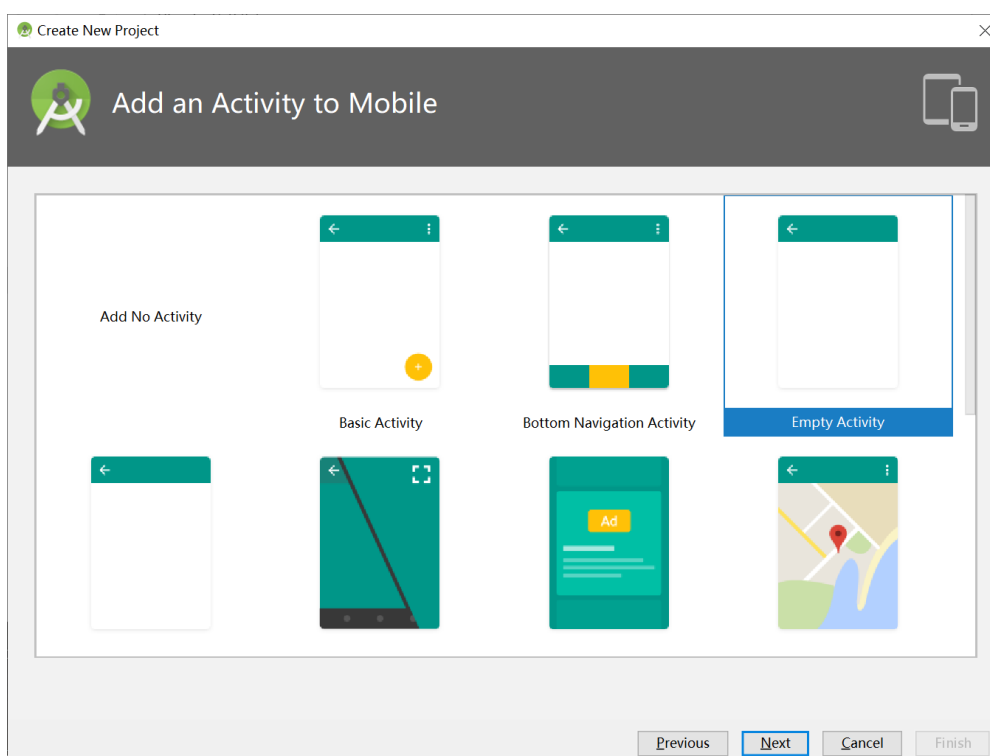
- 2) 在弹出的页面中填写，工程名称，此处为 “My Application”，在 Project location 下设置工程路径，其它保持默认即可。完成后点击 “Next” 按钮继续。



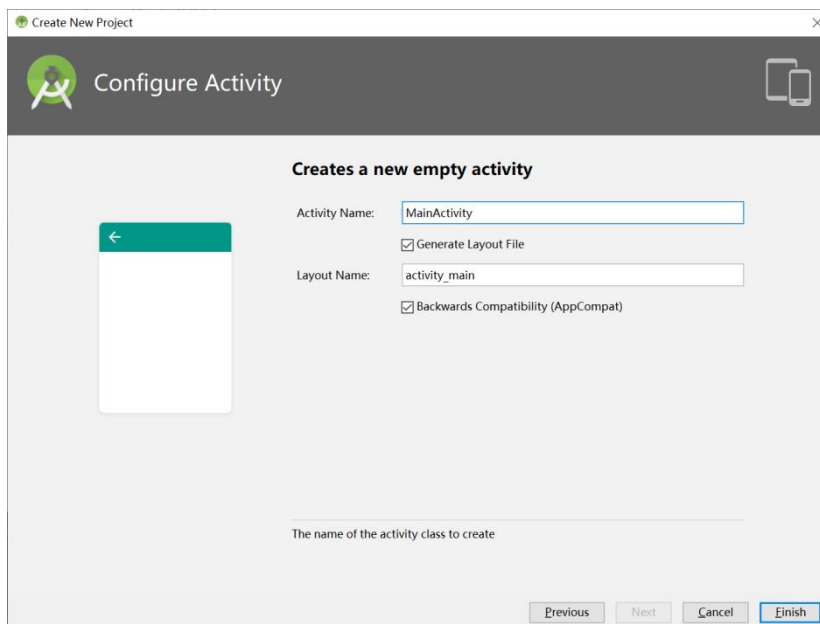
- 3) 随后弹出平台选择页面，目前 SDK 仅支持 Phone and Tablet，Android Things 平台，Android API 支持 6.0 以上版本。请选择满足要求的平台和 API 版本。正确选择完成后，点击 “Next” 按钮继续。



- 4) 进入添加 Activity 页面，在此选择一个需要的 Activity 模板即可，此处使用默认选项，选择 Empty Activity。确定好模板后，点击 Next 继续。

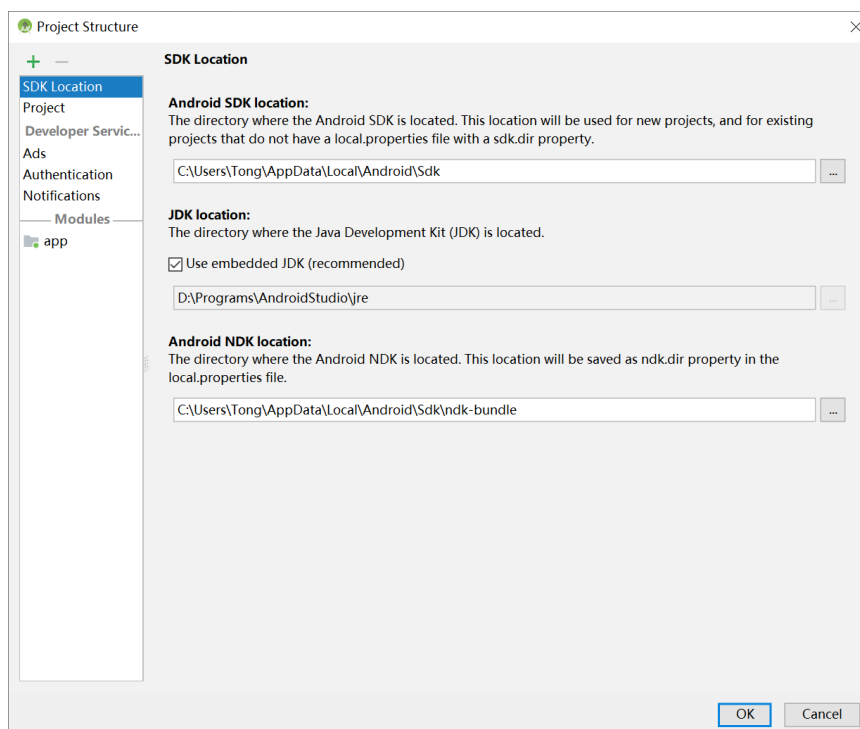


- 5) 接下来进入配置 Activity 页面，可修改 Activity 和 Layout 的名称修改名称，此处保持默认。点击 Finish 即可完成工程的创建。

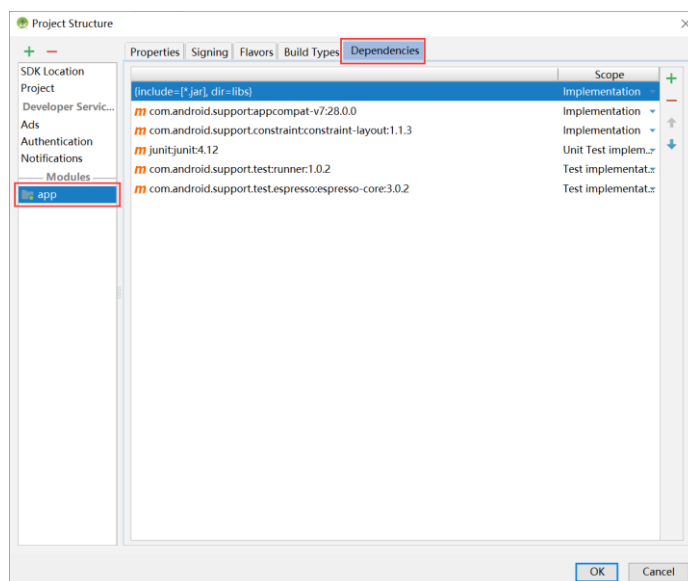


2.1.2 库的导入

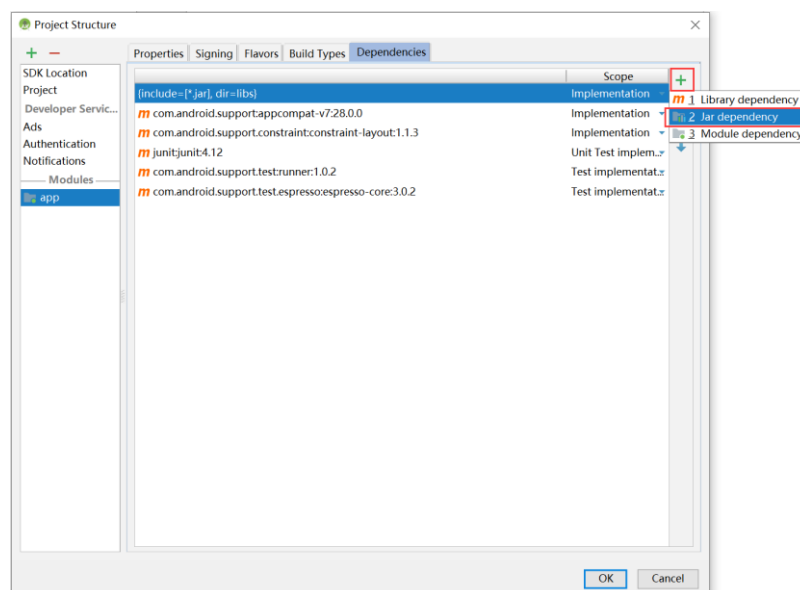
- 1) 将 SDK 包中 “\lib\galaxy-android-source\” 目录下 “galaxy-android-source.jar” 文件拷贝至工程目录下的库目录中（“MyApplication\app\libs”）；
- 2) 将 SDK 包中 “\lib\galaxy-android-jniLibs” 目录下 “jniLibs” 目录拷贝至工程目录下的 main 目录下（“MyApplication\app\src\main\”）；
- 3) 完成库的拷贝后，依次点击 “File” -> “Project Structure...”，或使用快捷键 “Ctrl + Alt + Shift + s”，进入工程结构设置页，如下图所示：



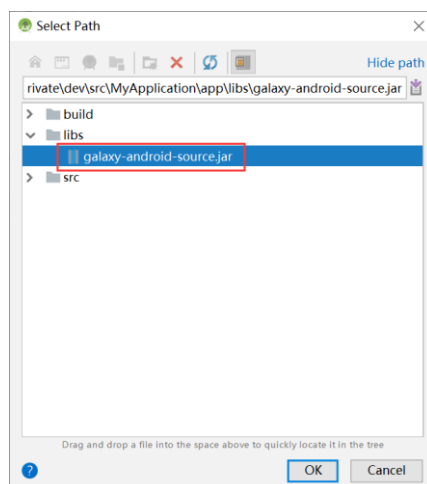
4) 在工程结构配置页面中依次选择 “app->Dependencies” ，出现如下图所示页面：



5) 点击页面右侧加号，出现如下图所示 3 个选项，选择 “2 Jar dependency” ；



6) 选择后，弹出如下图所示对话框：



7) 选择 libs 目录下的 “galaxy-android-source.jar” 包，随后点击 OK 完成选择；

8) 在 “Project Structure” 页面中继续点击 OK，完成库的导入。

2.1.3 javadoc 的配置

javadoc 作用是引入接口的注释说明，可以在代码编写时看到每个接口的注释说明，根据需求进行配置。

配置过程如下：

- 1) 将 SDK 包中 “\lib\galaxy-android-javadoc” 目录下 “galaxy-android-javadoc.jar” 文件拷贝至工程目录下的库目录中（ “MyApplication\app\libs” ）；
- 2) 在工程窗口中目录（ “MyApplication\idea\libraries” ）下找到引入的 “galaxy-android-source.jar” 所对应的 xml 文件，如下图所示：



3) 打开该 xml 文件，修改 “<JAVADOC />” 标签为：

```
<JAVADOC>
  <root url="jar://$PROJECT_DIR$/app/libs/galaxy-android-javadoc.jar!/" />
</JAVADOC>
```

修改完成后，如下图所示：



3 编程指引

3.1 引入库

为使用接口库，在程序的开始引入库。

示例代码：

```
// 导入设备库中的设备管理类
import galaxy.DeviceManager;

// 获取设备管理实例
DeviceManager deviceManager = DeviceManager.getInstance();
```

3.2 枚举设备

用户通过调用枚举接口枚举当前所有可用设备，可以取得设备信息列表 List<DeviceInfo> devInfoList。设备信息列表的元素个数为枚举到的设备个数，列表中元素的数据类型为 DeviceInfo，DeviceInfo 存放设备信息详见下表：

信息项	获取方法	类型
设备索引	getIndex()	整型
厂商名称	getVendorName()	字符串
设备型号	getModelName()	字符串
设备序列号	getSN()	字符串
设备显示名称	getDisplayName()	字符串
设备标识	getDeviceID()	字符串
用户自定义名称	getUserID()	字符串
权限状态	getAccessStatus()	AccessStatus
设备类	getDeviceClass()	DeviceClass
MAC 地址（GEV 相机特有）	getMAC()	字符串
IP 地址（GEV 相机特有）	getIP()	字符串
子网掩码（GEV 相机特有）	getSubnetMask()	字符串
网关（GEV 相机特有）	getGateway()	字符串
此设备连接的本机网卡 MAC（GEV 相机特有）	getNICMAC()	字符串
此设备连接的本机网卡 IP（GEV 相机特有）	getNICIP()	字符串
此设备连接的本机网卡子网掩码（GEV 相机特有）	getNICSubnetMask()	字符串
此设备连接的本机网卡网关（GEV 相机特有）	getNICGateway()	字符串

此设备连接的本机网卡描述（GEV 相机特有）	getNICDescription()	字符串
------------------------	---------------------	-----

3.2.1 普通方式

示例代码：

```
// 获取设备管理实例
DeviceManager deviceManager= DeviceManager.getInstance();

// 枚举设备（枚举超时时间为 200ms）
List<DeviceInfo> devInfoList = deviceManager.updateDeviceList(200);
if (devInfoList.size() == 0)
{
    //提示没有枚举到设备
}
else
{
    // 进行枚举到设备后的处理，例如打开设备、开始采集等
}
```

注意：

除上面的枚举接口 [updateDeviceList](#)，[DeviceManager](#) 还提供了另一个普通枚举接口 [updateAllDeviceList](#)：

- 1) 对于非 GEV 相机来说，这两个枚举接口功能上是一样的；
- 2) 对 GEV 相机来说，枚举机制不一样：

updateAllDeviceList：使用全网枚举，能够枚举到局域网内的所有 GEV 相机；

updateDeviceList：使用子网枚举，只能枚举到局域网内的同一网段的 GEV 相机。

3.2.2 无 Root 权限方式

样例代码见 [1.4.2 节](#)

注意：

除 [1.4.2 节](#) 样例的开始枚举接口 [startUpdateDeviceList](#)，[DeviceManager](#) 还提供了另一个开始枚举的接口 [startUpdateAllDeviceList](#)：

- 1) 对于非 GEV 相机来说，这两个接口功能上是一样的；
- 2) 对 GEV 相机来说，枚举机制不一样：

startUpdateAllDeviceList：开始 GEV 相机的全网枚举过程和 U3V 相机的枚举过程，能够枚举到局域网内的所有 GEV 相机和 U3V 相机；

startUpdateDeviceList：开始 GEV 相机的子网枚举过程和 U3V 相机的枚举过程，只能枚举到局域网内的同一网段的 GEV 相机和 U3V 相机；

- 3) 只使用**非 U3V** 相机的情况, **必须**使用普通枚举方式;
- 4) **不支持在多个线程中同时开始无 Root 权限枚举的过程。**

3.3 打开设备

用户通过调用以下五种不同的方式打开设备:

- 1) Device [openDeviceBySN](#)(String strSN, [AccessMode](#) accessMode);
- 2) Device [openDeviceByUserID](#) (String strUserID, AccessMode accessMode);
- 3) Device [openDeviceByIndex](#)(int index, AccessMode accessMode);
- 4) Device [openDeviceByIP](#)(String strIP, AccessMode accessMode);
- 5) Device [openDeviceByMAC](#)(String strMAC, AccessMode accessMode);

其中:

strSN 为设备序列号;

strUserID 为用户自定义名称;

index 为设备索引 (1,2,3...);

strIP 为设备 IP 地址 (非 GEV 相机不可用);

strMAC 为设备 MAC 地址 (非 GEV 相机不可用);

注意:

最后两个函数只针对 GEV 相机使用。

用户可以调用 [Device](#) 提供的 [closeDevice](#) 接口来关闭设备, 释放所有设备资源。

示例代码:

```
// 打开设备
// 方法一
// 通过序列号打开设备,也可以直接传入 SN 的字符串, 例如"NC0150120001"
String strSN = devInfoList.get(0).getSN();
Device cam = deviceManager.openDeviceBySN(strSN);

// 方法二
// 通过用户 ID 打开设备
// String strUserID = devInfoList.get(0).getUserID();
// Device cam = deviceManager.openDeviceByUserID(strUserID);

// 方法三
// 通过索引打开设备, 也可以传入 1、2、3....., 但需要注意索引从 1 开始
// int index = devInfoList.get(0).getIndex();
// Device cam = deviceManager.openDeviceByIndex(index);
```

```
// 下面为只针对 GEV 相机使用的打开方式
// 方法四
// 通过 ip 地址打开设备
// String strIP= devInfoList.get(0).getIP() ;
// Device cam = deviceManager.openDeviceByIP(strIP,
//                                     EnumDefineSet.AccessMode.CONTROL);

// 方法五
// 通过 mac 地址打开设备
// String strMAC= devInfoList.get(0).getMAC()
// Device cam = deviceManager.openDeviceByMAC(strMAC,
//                                     EnumDefineSet.AccessMode.CONTROL);

// 关闭设备
cam.closeDevice();
```

3.4 采集控制

在打开设备并设置好相机采集参数后，用户可调用 [Device.streamOn\(\)](#)和 [Device.streamOff\(\)](#)执行开停采。

可以通过 [Device.getStreamChannelNum\(\)](#)接口获得设备的流通道个数，当前库支持的所有相机都**只有一个流通道**。

可以通过 [Device.getDataStream\(0\)](#)获取第一个流通道 [DataStream](#) 类的对象。

示例代码：

```
// 获取 DataStream 类对象
DataStream dataStream = cam.getDataStream(0);

// 设置采集图像缓存 Buffer 个数,不设置的情况下默认为 5，需要在开采之前设置
dataStream.setAcquisitionBufferNumber(10);

// 开始采集
cam.streamOn();

// 图像采集 .....，具体见 3.5 节

// 使用 getRawImage 或 getBitmap 的采集的调用，如果需要每次调用的是最新采集的图像，
// 可以在调用 getRawImage 或 getBitmap 之前，先调用 dataStream.flushQueue() 即可
```

```
// 停止采集  
cam.streamOff();
```

3.5 图像采集与处理

[DataStream](#)类提供了 [getRawImage](#)/[getBitmap](#)/[getImageBySurface](#) 三个采集图像的接口，用户可以根据需要进行选择一种使用，注意不能同时使用。

3.5.1 getRawImage 方法

3.5.1.1 Raw 图的采集

获取 Raw 格式图像接口为 [getRawImage](#)，返回值为 [RawImage](#) 类的对象。

示例代码：

```
RawImage rawImage = null;  
try{  
    // 参数是超时时间，单位是 ms  
    rawImage = dataStream.getRawImage(500);  
}  
catch(ExceptionSet.Timeout timeout) {  
    // 捕获到超时，持续采集时直接继续，无需打印错误信息  
    continue;  
}  
catch(ExceptionSet exceptionSet) {  
    // 捕获到其他异常，打印异常信息后，持续采集时继续尝试获取图像或者停止  
    exceptionSet.printStackTrace();  
    // continue;  
}  
  
if(rawImage != null){  
    // 获取 Raw 图像数据  
    byte[] rawData = rawImage.getData();  
  
    // 根据需要进行图像数据的处理  
  
    // 保存 Raw 图  
    rawImage.save("Img0.raw");  
}
```

3.5.1.2 从 Raw 图转换 RGB

3.5.1.2.1 可选方式一

从 Raw 格式图像转换 RGB 格式图像调用的接口为 [convertToRGB](#)，返回的 [RGBImage](#) 类对象。

示例代码：

```
// 从 Raw 格式图像转换出 RGB 格式图像
RGBImage rgbImage = rawImage.convertToRGB(
    BayerConvertType.NEIGHBOUR,    // 邻域法 Bayer 转换
    ValidBit.BIT0_7),              // 图像格式为非 8 位，取 0~7 位
    false);                        // false 为不翻转

// 在 ARGB 数据基础上的图像质量提升，此一步用户可以根据需求选择是否进行
// 获取图像质量提升的参数 Gamma
ByteBuffer gammaLut = null;
if (cam.GammaParam.isReadable()) {
    double gammaValue = cam.GammaParam.get();
    gammaLut = Utility.getGammaLut(gammaValue);
}
else {
    // 当相机不支持推荐 Gamma 值的获取，可以有两种处理方式：
    // 1.gammaLut = null; 此种方式是不进行 Gamma 调节

    // 2.根据需求给定一个 Gamma 值，然后转换出 gammaLut。
}

// 获取图像质量提升的参数 Contrast
ByteBuffer contrastLut = null;
if (cam.ContrastParam.isReadable()) {
    long contrastValue = cam.ContrastParam.get();
    contrastLut = Utility.getContrastLut(contrastValue);
}
else {
    // 当相机不支持推荐的 Contrast 值的获取，可以有两种处理方式：
    // 1.contrastLut = null; 此种方式是不进行 Contrast 调节

    // 2.根据需求给定一个 Contrast 值，然后转换出 contrastLut。
}

// 获取颜色校正参数
long colorCorrectionParam = cam.ColorCorrectionParam.get();

// 实现图像质量提升
rgbImage.imageImprovement(colorCorrectionParam, contrastLut, gammaLut);
```

```
// 获取 RGB 格式图像数据
byte[] rgbData = rgbImage.getData();

// 根据需要进行 rgbData 数据进行处理 .....

// 还可以从 RGB 数据转换成 ARGB 数据, 然后得到 Bitmap
ARGBImage argbImage = rgbImage.convertToARGB((byte) 0xFF); // 参数为 Alpha 通道值

// 从 ARGBImage 对象中获取 Bitmap
Bitmap bitmap = argbImage.getBitmap();
```

3.5.1.2.2 可选方式二

从 Raw 格式图像转换 RGB 格式图像调用的接口为 [convert](#), 返回的 byte 数组。

示例代码:

```
// 从 Raw 格式图像转换出 RGB 格式图像

// 创建转换句柄
ImageFormatConvert imageConvert = new ImageFormatConvert();
imageConvert.createImageFormatConvertHandle();

// 设置插值方法
imageConvert.setInterpolationType(BayerConvertType.WEIGHT);

// 设置有效位, 图像格式为非 8 位, 取 0~7 位
imageConvert.setValidBits(ValidBit.BIT0_7);

// 设置目标格式
imageConvert.setDstFormat(PixelFormatEntry.RGB8);

// 图像宽高和输入 buffer 大小
int nWidth = rawImage.getWidth();
int nHeight = rawImage.getHeight();
int srcSize = nWidth * nHeight;

// 获取 RGB 的 buffer 大小
int dstSize = imageConvert.getBufferSizeForConversion(nWidth, nHeight,
    PixelFormatEntry.RGB8);

// RGB 的 buffer 申请内存
byte[] byRGBData = new byte[dstSize];
```

```
// Raw 格式转 RGB 格式
imageConvert.convert(rawImage.getData(),
                    srcSize,
                    byRGBData,
                    dstSize,
                    PixelFormatEntry.BAYER_RG8,
                    nWidth,
                    nHeight,
                    false);

// 根据需要进行 RGBData 数据进行处理 .....
// 比如进行锐化
sharpen24B(byRGBData, byRGBData, nWidth, nHeight, sharpness);

// 还可以从 RGB 数据得到 Bitmap
// 设置目标格式
imageConvert.setDstFormat(PixelFormatEntry.ARGB8);

// 输入、输出 buffer 大小
srcSize = nWidth * nHeight * 3;
dstSize = imageConvert.getBufferSizeForConversion(nWidth, nHeight,
                                                  PixelFormatEntry.ARGB8);

Bitmap bitmap = Bitmap.createBitmap(nWidth,
                                    nHeight,
                                    Bitmap.Config.ARGB_8888);

// 转换得到 Bitmap 图像
imageConvert.convert(byRGBData,
                    srcSize,
                    bitmap,
                    dstSize,
                    PixelFormatEntry.RGB8,
                    nWidth,
                    nHeight,
                    false);

// 销毁转换句柄
imageConvert.destroyImageFormatConvertHandle();
```

3.5.1.3 从 Raw 图转换 ARGB

3.5.1.3.1 可选方式一

从 Raw 格式图像转换 ARGB 格式图像调用的接口为 [convertToARGB](#)，返回的 [ARGBImage](#) 对象。

示例代码:

```
// 从 Raw 格式图像转换出 ARGB 格式图像
ARGBImage argbImage = rawImage.convertToARGB(
    BayerConvertType.NEIGHBOUR,    // 邻域法 Bayer 转换
    ValidBit.BIT0_7),              // 图像格式为非 8 位, 取 0~7 位
    false,                          // false 为不翻转
    (byte) 0xFF);                  // Alpha 通道值为 0xFF

// 在 ARGB 数据基础上的图像质量提升, 此一步用户可以根据需求选择是否进行
// 获取图像质量提升的参数 Gamma
ByteBuffer gammaLut = null;
if (cam.GammaParam.isReadable()) {
    double gammaValue = cam.GammaParam.get();
    gammaLut = Utility.getGammaLut(gammaValue);
}
else {
    // 当相机不支持推荐 Gamma 值的获取, 可以有两种处理方式:
    // 1. gammaLut = null; 此种方式是不进行 Gamma 调节

    // 2. 根据需求给定一个 Gamma 值, 然后转换出 gammaLut。
}

// 获取图像质量提升的参数 Contrast
ByteBuffer contrastLut = null;
if (cam.ContrastParam.isReadable()) {
    long contrastValue = cam.ContrastParam.get();
    contrastLut = Utility.getContrastLut(contrastValue);
}
else {
    // 当相机不支持推荐的 Contrast 值的获取, 可以有两种处理方式:
    // 1. contrastLut = null; 此种方式是不进行 Contrast 调节

    // 2. 根据需求给定一个 Contrast 值, 然后转换出 contrastLut。
}

// 获取颜色校正参数
long colorCorrectionParam = cam.ColorCorrectionParam.get();

// 实现图像质量提升
argbImage.imageImprovement(colorCorrectionParam, contrastLut, gammaLut);

// 获取 ARGB 格式图像数据
```



```
int[] argbData = argbImage.getData();

// 根据需要进行 argbData 数据进行处理 .....

// 从 ARGBImage 对象中获取 Bitmap
Bitmap bitmap = argbImage.getBitmap();
```

3.5.1.3.2 可选方式二

从 Raw 格式图像转换 ARGB 格式图像调用的接口为 [convert](#)，返回的 Bitmap 对象。

示例代码：

```
// 从 Raw 格式图像转换出 ARGB 格式图像

// 创建转换句柄
ImageFormatConvert imageConvert = new ImageFormatConvert();
imageConvert.createImageFormatConvertHandle();

// 设置插值方法
imageConvert.setInterpolationType(BayerConvertType.WEIGHT);

// 设置有效位，图像格式为非 8 位，取 0~7 位
imageConvert.setValidBits(ValidBit.BIT0_7);

// 设置目标格式
imageConvert.setDstFormat(PixelFormatEntry.ARGB8);

// 图像宽高和输入 buffer 大小
int nWidth = rawImage.getWidth();
int nHeight = rawImage.getHeight();
int srcSize = nWidth * nHeight;

// 获取 ARGB 的 buffer 大小
int dstSize = imageConvert.getBufferSizeForConversion(nWidth, nHeight,
    PixelFormatEntry.ARGB8);

// 设置目标格式
imageConvert.setDstFormat(PixelFormatEntry.ARGB8);

// 输入、输出 buffer 大小
srcSize = nWidth * nHeight * 3;
dstSize = imageConvert.getBufferSizeForConversion(nWidth, nHeight,
    PixelFormatEntry.ARGB8);

Bitmap bitmap = Bitmap.createBitmap(nWidth,
    nHeight,
```

```
Bitmap.Config.ARGB_8888);  
  
// 转换得到 Bitmap 图像  
imageConvert.convert(byRGBData,  
                    srcSize,  
                    bitmap,  
                    dstSize,  
                    PixelFormatEntry.BAYER_RG8,  
                    nWidth,  
                    nHeight,  
                    false);  
  
// 销毁转换句柄  
imageConvert.destroyImageFormatConvertHandle();
```

3.5.2 getBitmap 方法

[getBitmap](#) 方法是 [DataStream](#) 类提供的一种采集方式，主要实现从相机采集一幅 Raw 格式图像转换成 Bitmap 输出，返回值为 Bitmap 类型；

第一个参数类型为 ProcessParam，主要包含图像转换和图像质量提升的相关参数，具体见下表：

参数名称	参数类型	默认值	参数说明
bayerConvertType	BayerConvertType	NEIGHBOUR	Bayer 格式图像转换为 RGB 方法，默认邻域法
validBit	ValidBit	BIT0_7	将非 8 位 Raw 格式图像转换为 8 位的有效位选择
flip	boolean	false	是否翻转
mirror	boolean	false	是否进行图像镜像的处理
imageMirrorMode	ImageMirrorMode	HORIZONTAL_MIRROR	镜像模式，需要将 mirror 设置为 true，此参数才生效
colorCorrectionParam	long	0	颜色校正参数，0 为不处理
gammaLut	ByteBuffer	null	Gamma 查找表，null 为不处理
contrastLut	ByteBuffer	null	对比度查找表，null 为不处理
alpha	Byte	0xFF	Alpha 通道值

第二个参数为输出参数，类型为 FrameInfo，主要包含采集到图像的基本信息，具体见下表：

信息名称	类型	说明
status	int	图像状态，状态含义参考 FrameStatus ，当状态值是非 0 时， getBitmap 接口返回 null

width	int	图像宽度
height	int	图像高度
offsetX	int	获取图像 OffsetX
offsetY	int	获取图像 OffsetY
frameID	long	帧 ID
timestamp	long	时间戳

第三个参数为超时时间，单位为 ms，是每次调用该接口，阻塞的最长时间。

示例代码：

```
Bitmap bitmap = null;
ProcessParam processParam = new ProcessParam();
FrameInfo frameInfo = new FrameInfo();

// 下面是将图像采集与处理相关参数修改为非默认值的代码，如果默认值能满足需要，则不进行
// processParam.setBayerConvertType(
//     EnumDefineSet.BayerConvertType.NEIGHBOUR); // 插值算法设为边缘自适应
// processParam.setValidBit(
//     EnumDefineSet.ValidBit.BIT1_8); // 有效位选择，相机
PixelFormat 设置为非 8 位时有效
// processParam.setFlip(true); // 插值的同时进行上下翻转
// processParam.setMirror(true); // 进行镜像处理
// processParam.setImageMirrorMode(
//     EnumDefineSet.ImageMirrorMode.HORIZONTAL_MIRROR); // 水平镜像

// 初始化图像质量提升参数（如果不需要图像质量提升，这一步可以不做）
// 获取图像质量提升的参数 Gamma
if (cam.GammaParam.isReadable()) {
    double gammaValue = cam.GammaParam.get();
    ByteBuffer gammaLut = Utility.getGammaLut(gammaValue);
    processParam.setGammaLut(gammaLut);
}
else {
    // 当相机不支持推荐 Gamma 值的获取，可以有两种处理方式：
    // 1.gammaLut = null; 此种方式是不进行 Gamma 调节

    // 2.根据需求给定一个 Gamma 值，然后转换出 gammaLut。
}

// 获取图像质量提升的参数 Contrast
```

```
if (cam.ContrastParam.isReadable()) {
    long contrastValue = cam.ContrastParam.get();
    ByteBuffer contrastLut = Utility.getContrastLut(contrastValue);
    processParam.setContrastLut(contrastLut);
}
else {
    // 当相机不支持推荐的 Contrast 值的获取，可以有两种处理方式：
    // 1.contrastLut = null; 此种方式是不进行 Contrast 调节

    // 2.根据需求给定一个 Contrast 值，然后转换出 contrastLut。
}

// 获取颜色校正参数
processParam.setColorCorrectionParam(cam.ColorCorrectionParam.get());

try {
    // 参数是超时时间，单位是 ms
    bitmap = dataStream.getBitmap(processParam, frameInfo, 500);
}
catch (ExceptionSet.Timeout timeout) {
    // 捕获到超时，持续采集时直接继续，无需打印错误信息
    continue;
}
catch (ExceptionSet exceptionSet) {
    // 捕获到其他异常，打印异常信息后，持续采集时继续尝试获取图像或者停止
    exceptionSet.printStackTrace();
    // continue;
}

if (bitmap != null && frameInfo.getStatus() == 0) {
    // 根据需要对 bitmap 图像进行处理或显示
}
```

3.5.3 getImageBySurface 方法

[getImageBySurface](#)方法是 [DataStream](#) 类提供了一种采集方式，主要实现从相机采集一幅 Raw 格式图像，经过转换和处理后，将图像直接显示在控件（Surface）上的功能；

第一个参数的类型为 Surface，传入需要显示图像的控件类对象；

第二个参数的类型为 ProcessParam，主要包含图像转换和图像质量提升的相关参数，具体见下表：

参数名称	参数类型	默认值	参数说明
------	------	-----	------

bayerConvertType	BayerConvertType	NEIGHBOUR	Bayer 格式图像转换为 RGB 方法，默认邻域法
validBit	ValidBit	BIT0_7	将非 8 位 Raw 格式图像转换为 8 位的有效位选择
flip	boolean	false	是否翻转
mirror	boolean	false	是否进行图像镜像的处理
imageMirrorMode	ImageMirrorMode	HORIZONTAL_MIRROR	镜像模式，需要将 mirror 设置为 true，此参数才生效
colorCorrectionParam	long	0	颜色校正参数，0 为不处理
gammaLut	ByteBuffer	null	Gamma 查找表，null 为不处理
contrastLut	ByteBuffer	null	对比度查找表，null 为不处理
alpha	Byte	0xFF	Alpha 通道值

第三个参数为输出参数，类型为 FrameInfo，主要包含采集到图像的基本信息，具体见下表：

信息名称	类型	说明
status	int	图像状态，状态含义参考 FrameStatus
width	int	图像宽度
height	int	图像高度
offsetX	int	获取图像 OffsetX
offsetY	int	获取图像 OffsetY
frameID	long	帧 ID
timestamp	long	时间戳

第四个参数是保存图像的路径和名称，传入 null 时，不保存图像；

第五个参数为超时时间，单位为 ms，是每次调用该接口，未采集到图像阻塞的最长时间。

示例代码：

```
ProcessParam processParam = new ProcessParam();
FrameInfo frameInfo = new FrameInfo();

// 下面是将图像采集与处理相关参数修改为非默认值的代码，如果默认值能满足需要，则不进行
// processParam.setBayerConvertType(
//     EnumDefineSet.BayerConvertType.NEIGHBOUR); // 插值算法设为边缘自适应
// processParam.setValidBit(
//     EnumDefineSet.ValidBit.BIT1_8); // 有效位选择，相机
PixelFormat 设置为非 8 位时有效
```

```
// processParam.setFlip(true); // 插值的同时进行上下翻转
// processParam.setMirror(true); // 进行镜像处理
// processParam.setImageMirrorMode(
    EnumDefineSet.ImageMirrorMode.HORIZONTAL_MIRROR); // 水平镜像

// 初始化图像质量提升参数（如果不需要图像质量提升，这一步可以不做）
// 获取图像质量提升的参数 Gamma
if (cam.GammaParam.isReadable()) {
    double gammaValue = cam.GammaParam.get();
    ByteBuffer gammaLut = Utility.getGammaLut(gammaValue);
    processParam.setGammaLut(gammaLut);
}
else {
    // 当相机不支持推荐 Gamma 值的获取，可以有两种处理方式：
    // 1.gammaLut = null; 此种方式是不进行 Gamma 调节

    // 2.根据需求给定一个 Gamma 值，然后转换出 gammaLut。
}

// 获取图像质量提升的参数 Contrast
if (cam.ContrastParam.isReadable()) {
    long contrastValue = cam.ContrastParam.get();
    ByteBuffer contrastLut = Utility.getContrastLut(contrastValue);
    processParam.setContrastLut(contrastLut);
}
else {
    // 当相机不支持推荐的 Contrast 值的获取，可以有两种处理方式：
    // 1.contrastLut = null; 此种方式是不进行 Contrast 调节

    // 2.根据需求给定一个 Contrast 值，然后转换出 contrastLut。
}

// 获取颜色校正参数
processParam.setColorCorrectionParam(cam.ColorCorrectionParam.get());

try {
    // 尝试采集一幅图像并显示在界面
    dataStream.getImageBySurface(
        surface,
        processParam,
        frameInfo,
        null, // 保存 Raw 图像路径，传 null 不存图
        500); //单位是 ms
}
```

```

}
catch (ExceptionSet.Timeout timeout) {
    // 捕获到超时，持续采集时直接继续，无需打印错误信息
    continue;
}
catch (ExceptionSet exceptionSet) {
    // 捕获到其他异常，打印异常信息后，持续采集时继续尝试获取图像或者停止
    exceptionSet.printStackTrace();
    // continue;
}
}

```

3.5.4 图像质量提升

Android 下 Java 接口库提供的三种图像采集接口都可以进行图像质量提升。图像质量提升功能主要包括颜色校正、Gamma、对比度等图像处理的操作，用户可以有选择的进行。具体的用法见上文中三种采集方法的示例代码。

1) 颜色校正

设备属性参数名称：ColorCorrectionParam

名词解释：提高相机色彩还原度，使图像更加接近人眼视觉感受。

2) Gamma 调节

Gamma 值：浮点型，范围[0.1, 10.0]，缺省值为 1

设备属性参数名称：GammaParam

名词解释：Gamma 调节是为了让显示器的输出尽量接近输入。

3) 对比度调节

Contrast 值：整型，范围[-50, 100]，缺省值为 0

设备属性参数名称：ContrastParam

名词解释：图像明亮部分与黑暗部分的亮度比称为对比度。对比度高或者反差大的图像，其中被摄景物的轮廓较清楚，图像也较清晰；反之，对比度低的图像轮廓不清，图像也不太清晰。

3.6 相机控制

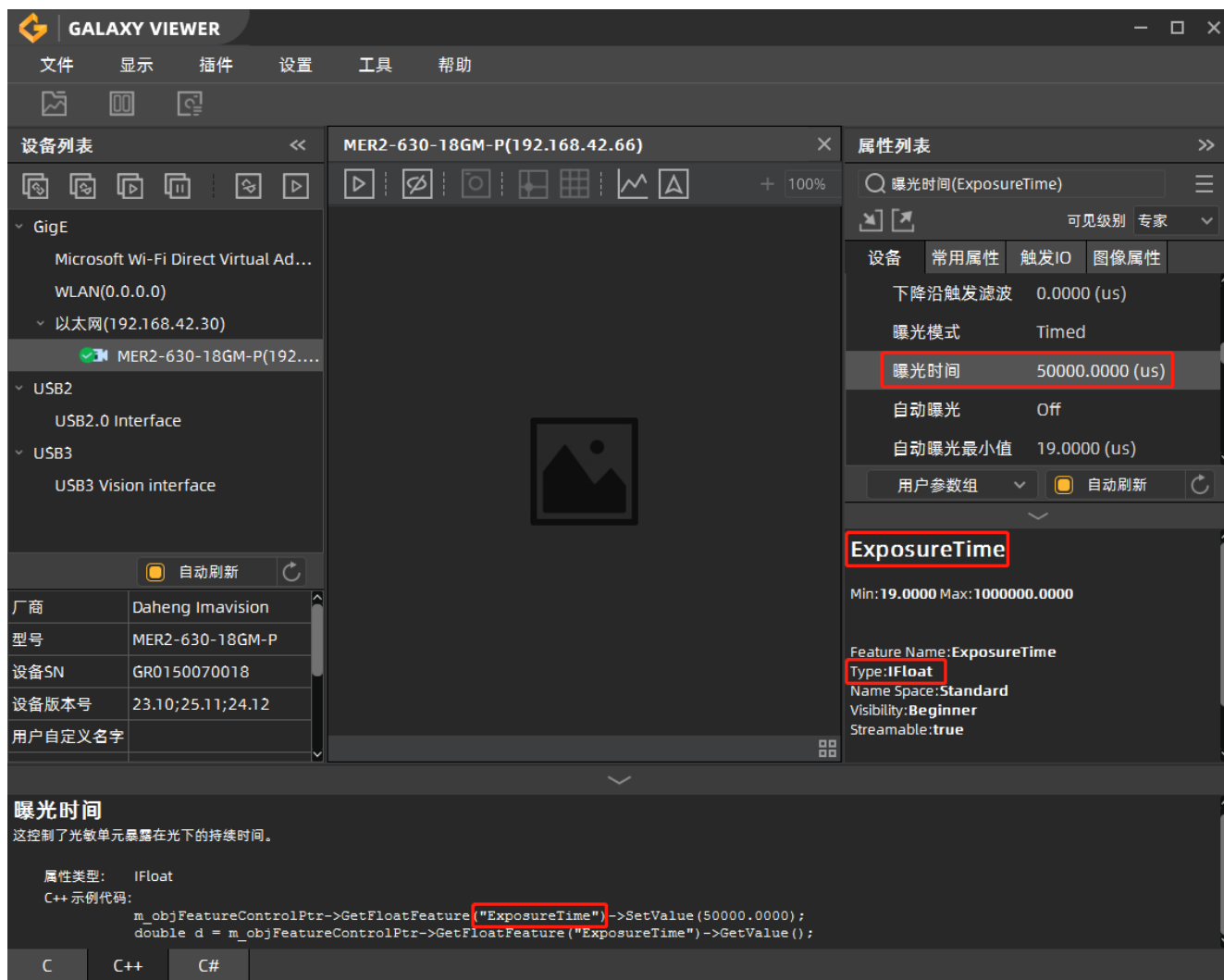
相机的控制功能做了一些拖拽，支持使用字符串的方式访问相机。

3.6.1 属性节点字符串模式访问

本次添加的功能在原有功能码设置的基础上，增加了字符串的属性设置功能。相机的所有功能均可采用字符串的属性设置方式实现，支持字符串功能查询，可读、可写查询，属性值的设置和获取。

3.6.1.1 属性对应字符串的查询

相机功能属性节点对应唯一的字符串命名，这个是保存到相机的 XML 文件里，也可以通过 Windows 的 Galaxy Viewer 示例程序查询到对应的字符串。如下图所示：



如上所示，可以从两个地方获取该属性对应的字符串信息和属性数据类型信息。

3.6.1.2 字符串设置使用

Device 设备类中新增了获取各基础类型的属性对象，列表如下：

FeatureSet.FloatFeature getFloatFeature(String)	获取浮点型属性对象
FeatureSet.IntFeature getIntFeature(String)	获取整型属性对象
FeatureSet.EnumFeature getEnumFeature(String)	获取枚举型属性对象
FeatureSet.BoolFeature getBoolFeature(String)	获取布尔型属性对象
FeatureSet.StringFeature getStringFeature(String)	获取字符串型属性对象
FeatureSet.BufferFeature getBufferFeature(String)	获取寄存器型属性对象
FeatureSet.CommandFeature getCommandFeature(String)	获取命令型属性对象
boolean isImplemented();	判断属性参数是否已实现

boolean isReadable();	判断属性参数是否可读
boolean isWritable ();	判断属性参数是否可写

示例代码：

```
//device 为打开设备后的 Device 对象

//获取曝光时间属性对象
FeatureSet.FloatFeature exposureTimeObj =
device.getFloatFeature("ExposureTime");

//判断曝光时间是否支持并且可读 可写
if (exposureTimeObj.isImplemented())
{
    //判断曝光时间是否可读
    if (exposureTimeObj.isReadable())
    {
        //获取曝光时间最小值
        double dMinValue = exposureTimeObj.getMin();

        //获取曝光时间最大值
        double dMaxValue = exposureTimeObj.getMax();

        //获取曝光时间当前值
        double dValue = exposureTimeObj.get();

        //获取曝光时间步长
        double dMin = exposureTimeObj.getInc();

        //获取曝光时间最小值
        String strUnit = exposureTimeObj.getUnit();
    }

    //判断曝光时间是否可写
    if (exposureTimeObj.isWritable())
    {
        exposureTimeObj.set(10000.0);
    }
}
```

示例代码：

```
//device 为打开设备后的 Device 对象
```

```
//获取用户参数组加载属性对象
```

```
FeatureSet.CommandFeature CommandFeature =  
    device.CommandFeature("UserSetLoad");
```

```
//发送命令
```

```
commandObj.sendCommand();
```

3.6.2 属性直接访问（旧接口）

Java 旧接口方式访问相机，对于一些新的功能节点，如 Sensor 宽高信息、环境光源预设、黑电平、PGA 增益、用户数据区操作、2D 降噪、多源触发、序列、平场矫正功能、Burst 功能、串口操作等，旧接口不在升级和维护，遇到这些属性设置时候，请使用[字符串模式访问](#)。

3.6.2.1 属性参数访问类型

属性参数的访问属性分三种类型：是否实现、是否可读、是否可写。接口设计如下：

boolean isImplemented();	当前属性控制器是否支持此功能
boolean isReadable();	此功能是否可读
boolean isWritable();	此功能是否可写

建议用户在操作属性参数前，先查询属性的访问类型。

示例代码：

```
// 获取功能是否实现
```

```
DataStream dataStream = cam.getDataStream(0);  
boolean isImplemented = dataStream.StreamTransferSize.isImplemented();  
if(isImplemented)  
{
```

```
    // 获取是否可写
```

```
    boolean isWritable = dataStream.StreamTransferSize.isWritable();  
    if(isWritable)  
    {  
        // 设置 URB 大小  
        dataStream.StreamTransferSize.set(512*1024);  
    }  
}
```

```
// 获取是否可读
```

```
boolean isReadable = dataStream.StreamTransferSize.isReadable();  
if(isReadable)  
{
```

```
    // 获取 URB 大小当前值
```

```
    long transferSize = dataStream.StreamTransferSize.get();
```

```
}  
}
```

3.6.2.2 属性控制

属性控制分为[设备属性参数](#)和[流属性参数](#)两类。区别在于属性参数的控制功能有所不同：

- 1) 设备属性参数：设备信息，比如宽高、曝光增益等；
- 2) 流属性参数：关于采集控制和采集数据统计的属性。

接口调用根据属性参数的类型的不同分类：

整型([IntFeature](#)):

相关接口：

<code>void set(long value);</code>	<code>//设置</code>
<code>long get();</code>	<code>//读取</code>
<code>long getMin();</code>	<code>//获取最小值</code>
<code>long getMax();</code>	<code>//获取最大值</code>
<code>long getInc();</code>	<code>//获取步长</code>

示例代码：

```
// 获取图像宽度最大值  
long max = cam.Width.getMax();  
  
// 设置当前图像宽度为最大值  
cam.Width.set(max);  
  
// 获取当前图像宽度  
long value = cam.Width.get();
```

浮点型([FloatFeature](#)):

相关接口：

<code>void set(double value);</code>	<code>//设置</code>
<code>double get();</code>	<code>//读取</code>
<code>double getMin();</code>	<code>//获取最小值</code>
<code>double getMax();</code>	<code>//获取最大值</code>
<code>double getInc();</code>	<code>//获取精度</code>

示例代码：

```
// 获取曝光值可设置最大值  
double max = cam.ExposureTime.getMax();
```

```
// 设置当前曝光值为 10ms  
cam.ExposureTime.set(10000);  
  
// 获取当前曝光值  
double curValue = cam.ExposureTime.get();
```

枚举型(EnumFeature):

相关接口:

Void	set(long enumValue);	//设置
long	get();	//读取
List<EnumDescription>	getRange();	//获取枚举范围的列表

示例代码:

```
// 获取枚举值可设置范围  
List<EnumDescription> enumDescList = cam.BalanceWhiteAuto.getRange();  
  
// 设置当前枚举值 (开启连续自动白平衡)  
cam.BalanceWhiteAuto.set(EnumDefineSet.AutoEntry.CONTINUOUS.getValue());  
  
// 获取当前自动白平衡状态  
long value = cam.BalanceWhiteAuto.get();
```

布尔型 (BoolFeature) :

相关接口:

void	set(boolean boolValue);	//设置
boolean	get();	//读取

示例代码:

```
// 设置当前布尔值  
cam.LineInverter.set(true);  
  
// 读取布尔值  
boolean lineInverterValue = cam.LineInverter.get();
```

字符串型 (StringFeature) :

相关接口:

void	set(String value);	//设置
String	get();	//读取
int	getMaxLength();	//获取字符串型属性参数的最大长度值

示例代码:

```
// 读取最长字符串长度
int maxLength = cam.DeviceUserID.getMaxLength();

// 读取当前字符串值
String curValue = cam.DeviceUserID.get();

// 设置字符串值
cam.DeviceUserID.set("cam0");
```

Buffer 型([BufferFeature](#)):

相关接口:

void	set(ByteBuffer data);	//设置
	ByteBuffer get();	//读取
int	getLength();	//获取 Buffer 型属性参数的长度

示例代码:

```
// 读取缓冲数据长度
int bufferLength = cam.LUTValueAll.getLength();

// 读取块数据 (LUT)
ByteBuffer bufferData = cam.LUTValueAll.get();

// 设置块数据 (LUT)
cam.LUTValueAll.set(bufferData);
```

Command 型 ([CommandFeature](#)) :

相关接口:

void	sendCommand();	//发送命令
------	----------------	--------

示例代码:

```
// 发送软触发命令
cam.TriggerSoftware.sendCommand();
```

3.7 导入导出相机配置

导出相机配置文件的功能实现了将相机所有可设置参数的当前值保存到一个本地文件中, 在需要的时候再导入到相机中, 用户可以导出多个配置文件, 然后根据需要导入不同的参数。

示例代码:

```
// 导出配置相机参数文件
cam.exportConfigFile("config0.txt");

// 导入配置相机参数文件
cam.importConfigFile("config0.txt", false);
```

3.8 异常处理

当调用接口函数内部出现异常时，异常处理机制会检测并抛出不同类型异常，异常类型均继承自 ExceptionSet。

3.8.1 示例代码

一个典型的异常处理代码示例如下：

```
Device cam = null;
try
{
    // 调用接口函数时函数内部抛异常
    Device cam = deviceManager.openDeviceByIndex(1);
}
Catch(ExceptionSet exceptionSet)
{
    exceptionSet.printStackTrace();
}
```

用户也可通过捕获不同异常的具体错误类型，进行分类处理：

```
RawImage rawImage = null;
try{
    // 参数是超时时间，单位是 ms
    rawImage = dataStream.getRawImage(500);
}
catch(ExceptionSet.Timeout timeout) {
    // 捕获到超时，持续采集时直接继续，无需打印错误信息
    continue;
}
catch(ExceptionSet exceptionSet) {
    // 捕获到其他异常，打印异常信息后，持续采集时继续尝试获取图像或者停止
    exceptionSet.printStackTrace();
    // continue;
}
```

3.8.2 异常类型

异常类型	意义
UnexpectedError	不期望的异常
NotFoundTL	没找到 TL
NotFoundDevice	未找到设备
OffLine	相机掉线
InvalidParameter	参数无效
InvalidHandle	句柄无效
InvalidCall	无效调用
InvalidAccess	无效访问
NeedMoreBuffer	Buffer 不足
FeatureTypeError	功能类型错误
OutOfRange	超过范围
NotImplemented	功能未实现
NotInitApi	未初始化
Timeout	超时
ParameterTypeError	参数类型错误
RepeatOpened	重复打开
NotSupported	不支持的功能
PixelFormatNotSupport	不支持的像素格式
OutOfMemory	资源耗尽
OnlySupportAndroid	仅 Android 支持
SaveImageError	保存图像错误
SaveImageNoPermission	没有保存图像的权限
BitmapError	操作 bitmap 错误
CpuNotSupportAccelerate	Cpu 不支持图像处理库加速
NativeWindowError	操作 surface 错误

4 常见问题解答

序号	常见问题	解决办法
1.	Android 枚举不到 U3V 设备	使用无 Root 权限的枚举方式
2.	Android Studio 开发环境如何显示每个接口的提示说明?	导入 Javadoc, 具体方法见 2.1.3 节
3.	Android Studio 开发的工程程序, 安装或者 Debug 提示时候, 无法找到 libgxiapi.so	推荐查看《Android 版 SDK 常见问题解答》之问题五

5 附录

5.1 属性参数

注：MER-GEV：水星一代 GEV 相机；MER-U3V：水星一代 USB3.0 相机；MER-U2：水星一代 USB2.0 相机；MER2-GEV：水星二代 GEV 相机。

5.1.1 设备属性参数

属性参数	解释	属性类	
DeviceInformation Section			MER-U3V
DeviceVendorName	厂商名称	StringFeature	√
DeviceModelName	设备型号	StringFeature	√
DeviceFirmwareVersion	设备固件版本	StringFeature	√
DeviceVersion	设备版本	StringFeature	√
DeviceSerialNumber	设备序列号	StringFeature	√
FactorySettingVersion	出厂参数版本	StringFeature	√
DeviceUserID	用户自定义名称	StringFeature	√
DeviceLinkSelector	设备链路选择，详见 C 软件开发说明书	IntFeature	√
DeviceLinkThroughputLimit	设备链路带宽限制	IntFeature	√
DeviceLinkThroughputLimitMode	设备带宽限制模式，详见 SwitchEntry	EnumFeature	√
DeviceLinkCurrentThroughput	当前设备采集带宽	IntFeature	√
DeviceReset	设备复位	CommandFeature	
TimestampTickFrequency	时间戳时钟频率	IntFeature	
TimestampLatch	时间戳锁存	CommandFeature	
TimestampReset	重置时间戳	CommandFeature	
TimestampLatchReset	重置时间戳锁存	CommandFeature	
TimestampLatchValue	时间戳锁存值	IntFeature	
ImageFormat Section			MER-U3V
SensorWidth	传感器宽度	IntFeature	√
SensorHeight	传感器高度	IntFeature	√

WidthMax	最大宽度	IntFeature	√
HeightMax	最大高度	IntFeature	√
OffsetX	水平偏移	IntFeature	√
OffsetY	垂直偏移	IntFeature	√
Width	图像宽度	IntFeature	√
Height	图像高度	IntFeature	√
BinningHorizontal	水平像素 Binning	IntFeature	√
BinningVertical	垂直像素 Binning	IntFeature	√
DecimationHorizontal	水平像素抽样	IntFeature	√
DecimationVertical	垂直像素抽样	IntFeature	√
PixelSize	像素位深, 详见 PixelSizeEntry	EnumFeature	√
PixelColorFilter	Bayer 格式, 详见 PixelColorFilterEntry	EnumFeature	√
PixelFormat	像素格式, 详见 PixelFormatEntry	EnumFeature	√
ReverseX	水平翻转	BoolFeature	√
ReverseY	垂直翻转	BoolFeature	√
TestPattern	测试图, 详见 TestPatternEntry	EnumFeature	√
TestPatternGeneratorSelector	测试图源选择, 详见 C 软件开发说明书和 TestPatternGeneratorSelectorEntry	EnumFeature	√
RegionSendMode	ROI 输出模式, 详见 RegionSendModeEntry	EnumFeature	√
RegionMode	区域开关, 详见 SwitchEntry	EnumFeature	√
RegionSelector	区域选择, 详见 C 软件开发说明书和 RegionSelectorEntry	EnumFeature	√
CenterWidth	窗口宽度	IntFeature	√ *
CenterHeight	窗口高度	IntFeature	√ *
BinningHorizontalMode	水平像素 Binning 模式, 详见 BinningHorizontalModeEntry	EnumFeature	
BinningVerticalMode	垂直像素 Binning 模式, 详见 BinningVerticalModeEntry	EnumFeature	

SensorShutterMode	Sensor 曝光模式, 详见 SensorShutterModeEntry	EnumFeature	
TransportLayer Section			MER-U3V
PayloadSize	数据大小	IntFeature	√
CenterWidth	窗口宽度	IntFeature	√
CenterHeight	窗口高度	IntFeature	√
GevCurrentIPConfigurationLLA	LLA 方式配置 IP	BoolFeature	
GevCurrentIPConfigurationDHCP	DHCP 方式配置 IP	BoolFeature	
GevCurrentIPConfigurationPersistentIP	永久 IP 方式配置 IP	BoolFeature	
EstimatedBandwidth	预估带宽	IntFeature	
GevHeartbeatTimeout	心跳超时时间	IntFeature	
GevSCPSPacketSize	流通道包长	IntFeature	
GevSCPD	流通道包间隔	IntFeature	
GevLinkSpeed	连接速度	IntFeature	
DigitalIO Section			MER-U3V
UserOutputSelector	用户自定义输出选择, 详见 C 软件开发 说明书和 UserOutputSelectorEntry	EnumFeature	√
UserOutputValue	用户自定义输出值	BoolFeature	√
UserOutputMode	用户 IO 输出模式, 详见 UserOutputModeEntry	EnumFeature	
StrobeSwitch	闪光灯开关, 详见 SwitchEntry	EnumFeature	
LineSelector	引脚选择, 详见 C 软件开发说明书和 LineSelectorEntry	EnumFeature	√
LineMode	引脚方向, 详见 LineModeEntry	EnumFeature	√
LineSource	引脚输出源, 详见 LineSourceEntry	EnumFeature	√
LineInverter	引脚电平反转	BoolFeature	√
LineStatus	引脚状态	BoolFeature	√
LineStatusAll	所有引脚的状态	IntFeature	√
AnalogControls Section			MER-U3V

GainAuto	自动增益, 详见 AutoEntry	EnumFeature	√
GainSelector	增益通道选择, 详见 C 软件开发说明书和 GainSelectorEntry	EnumFeature	√
BlackLevelAuto	自动黑电平, 详见 AutoEntry	EnumFeature	√
BlackLevelSelector	黑电平通道选择, 详见 C 软件开发说明书和 BlackLevelSelectEntry	EnumFeature	√
BalanceWhiteAuto	自动白平衡, 详见 AutoEntry	EnumFeature	√
BalanceRatioSelector	白平衡通道选择, 详见 C 软件开发说明书和 BalanceRatioSelectorEntry	EnumFeature	√
BalanceRatio	白平衡系数	FloatFeature	√
DeadPixelCorrect	坏点校正	EnumFeature	√
Gain	增益	FloatFeature	√
BlackLevel	黑电平	FloatFeature	√
GammaEnable	Gamma 使能	BoolFeature	
GammaMode	Gamma 模式, 详见 GammaModeEntry	EnumFeature	
Gamma	Gamma	FloatFeature	
DigitalShift	数字移位	IntFeature	
CustomFeature Section			MER-U3V
ADCLLevel	AD 转换级别	IntFeature	
HBlanking	水平消隐	IntFeature	
VBlanking	垂直消隐	IntFeature	
UserPassword	用户加密区密码	StringFeature	
VerifyPassword	用户加密区校验密码	StringFeature	
UserData	用户加密区内容	BufferFeature	
ExpectedGrayValue	期望灰度值	IntFeature	√
AALightEnvironment	自动曝光、自动增益, 光照环境类型, 详见 AALightEnvironmentEntry	EnumFeature	
ImageGrayRaiseSwitch	图像亮度拉伸开关, 详见 SwitchEntry	EnumFeature	

AAROIOffsetX	自动调节感兴趣区域 X 坐标	IntFeature	√
AAROIOffsetY	自动调节感兴趣区域 Y 坐标	IntFeature	√
AAROIWidth	自动调节感兴趣区域宽度	IntFeature	√
AAROIHeight	自动调节感兴趣区域高度	IntFeature	√
AutoGainMin	自动增益最小值	FloatFeature	√
AutoGainMax	自动增益最大值	FloatFeature	√
AutoExposureTimeMin	自动曝光最小值	FloatFeature	√
AutoExposureTimeMax	自动曝光最大值	FloatFeature	√
ContrastParam	对比度参数	IntFeature	√
ColorCorrectionParam	颜色校正系数	IntFeature	√
AWBROIOffsetX	自动白平衡感兴趣区域 X 坐标	IntFeature	√
AWBROIOffsetY	自动白平衡感兴趣区域 Y 坐标	IntFeature	√
AWBROIWidth	自动白平衡感兴趣区域宽度	IntFeature	√
AWBROIHeight	自动白平衡感兴趣区域高度	IntFeature	√
GammaParam	伽马参数	FloatFeature	√
AWBLampHouse	自动白平衡光源, 详见 AWBLampHouseEntry	EnumFeature	√
SharpnessMode	锐化模式, 详见 SwitchEntry	EnumFeature	√
Sharpness	锐度	FloatFeature	√
FrameInformation	图像帧信息	BufferFeature	
UserSetControl Section			MER-U3V
UserSetLoad	加载参数组	CommandFeature	√
UserSetSave	保存参数组	CommandFeature	√
UserSetSelector	参数组选择, 详见 C 软件开发说明书和 UserSetEntry	EnumFeature	√
UserSetDefault	启动参数组, 详见 UserSetEntry	EnumFeature	√
LUT Section			MER-U3V

LUTValueAll	查找表内容	BufferFeature	√
LUTSelector	查找表选择, 详见 C 软件开发说明书和 LutSelectorEntry	EnumFeature	√
LUTEnable	查找表使能	BoolFeature	
LUTIndex	查找表索引	IntFeature	
LUTValue	查找表值	IntFeature	
Color Transformation Control			MER-U3V
ColorTransformationMode	颜色转换模式, 详见 ColorTransformationModeEntry	EnumFeature	
ColorTransformationEnable	颜色转换使能	BoolFeature	
ColorTransformationValueSelector	颜色转换矩阵元素选择, 详见 ColorTransformationValueSelectorEntry	EnumFeature	
ColorTransformationValue	颜色转换矩阵元素	FloatFeature	
ChunkData Section			MER-U3V
ChunkModeActive	帧信息使能	BoolFeature	√
ChunkEnable	单项帧信息使能	BoolFeature	√
ChunkSelector	帧信息项选择, 详见 C 软件开发说明书和 ChunkSelectorEntry	EnumFeature	√
Device Feature			MER-U3V
DeviceCommandTimeout	命令超时	IntFeature	
DeviceCommandRetryCount	命令重试次数	IntFeature	
AcquisitionControl Section			MER-U3V
FrameBufferOverwriteActive	帧存覆盖使能	BoolFeature	√
TriggerSoftware	软触发	CommandFeature	√
TransferStart	开始传输	CommandFeature	√
AcquisitionMode	采集模式, 详见 AcquisitionModeEntry	EnumFeature	√
TriggerMode	触发模式, 详见 SwitchEntry	EnumFeature	√
TriggerActivation	触发极性, 详见 TriggerActivationEntry	EnumFeature	√
ExposureAuto	自动曝光, 详见 AutoEntry	EnumFeature	√

TriggerSource	触发表, 详见 TriggerSourceEntry	EnumFeature	√
ExposureMode	曝光模式, 详见 ExposureModeEntry	EnumFeature	√
TriggerSelector	触发类型选择, 详见 C 软件开发说明书和 TriggerSelectorEntry	EnumFeature	√
TransferControlMode	传输控制模式, 详见 TransferControlModeEntry	EnumFeature	√
TransferOperationMode	传输操作模式, 详见 TransferOperationModeEntry	EnumFeature	√
AcquisitionFrameRateMode	采集帧率调节模式, 详见 SwitchEntry	EnumFeature	√
FixedPatternNoiseCorrectMode	模板噪声校正, 详见 SwitchEntry	EnumFeature	√
ExposureTime	曝光时间	FloatFeature	√
TriggerFilterRaisingEdge	上升沿触发滤波	FloatFeature	√
TriggerFilterFallingEdge	下降沿触发滤波	FloatFeature	√
TriggerDelay	触发延迟	FloatFeature	√
AcquisitionFrameRate	采集帧率	FloatFeature	√
CurrentAcquisitionFrameRate	当前采集帧率	FloatFeature	√
TransferBlockCount	传输帧数	IntFeature	√
TriggerSwitch	外触发开关, 详见 SwitchEntry	EnumFeature	
AcquisitionSpeedLevel	采集速度级别	IntFeature	
AcquisitionFrameCount	多帧采集帧数	IntFeature	
AcquisitionBurstFrameCount	高速连拍帧数	IntFeature	
AcquisitionStatusSelector	采集状态选择, 详见 C 软件开发说明书和 AcquisitionStatusSelectorEntry	EnumFeature	
AcquisitionStatus	采集状态	BoolFeature	
ExposureDelay	曝光延迟	FloatFeature	
CounterAndTimerControl Section			MER-U3V
TimerSelector	计时器选择, 详见 TimerSelectorEntry	EnumFeature	√
TimerDuration	计时器持续时间	FloatFeature	√
TimerDelay	计时器延迟	FloatFeature	√

TimerTriggerSource	计时器触发源, 详见 TimerTriggerSourceEntry	EnumFeature	√
CounterSelector	计数器选择, 详见 CounterSelectorEntry	EnumFeature	√
CounterEventSource	计数器事件触发源, 详见 CounterEventSourceEntry	EnumFeature	√
CounterResetSource	计数器复位源, 详见 CounterResetSourceEntry	EnumFeature	√
CounterResetActivation	计数器复位信号极性, 详见 CounterResetActivationEntry	EnumFeature	√
CounterReset	计数器复位	CommandFeature	√

CenterWidth、CenterHeight (√ *);仅支持基于 MER-U3V 的双目相机。

5.1.2 流属性参数

属性参数	解释	属性类	MER-U3V
StreamAnnouncedBufferCount	声明的 Buffer 个数	IntFeature	√
StreamDeliveredFrameCount	接收帧个数(包括残帧);	IntFeature	√
StreamLostFrameCount	buffer 不足导致的丢帧个数	IntFeature	√
StreamIncompleteFrameCount	接收的残帧个数	IntFeature	√
StreamDeliveredPacketCount	接收到的包数	IntFeature	√
StreamResendPacketCount	重传包个数	IntFeature	
StreamRescuedPacketCount	重传成功包个数	IntFeature	
StreamResendCommandCount	重传命令次数	IntFeature	
StreamUnexpectedPacketCount	异常包个数	IntFeature	
MaxPacketCountInOneBlock	数据块最大重传包数	IntFeature	
MaxPacketCountInOneCommand	一次重传命令最大包含的包数	IntFeature	
ResendTimeout	重传超时时间	IntFeature	
MaxWaitPacketCount	最大等待包数	IntFeature	
ResendMode	重传模式, 详见 SwitchEntry	EnumFeature	
StreamMissingBlockIDCount	BlockID 丢失个数	IntFeature	
BlockTimeout	数据块超时时间	IntFeature	

MaxNumQueueBuffer	采集队列最大 Buffer 个数	IntFeature	
PacketTimeout	包超时时间	IntFeature	
StreamTransferSize	传输数据块大小	IntFeature	√
StreamTransferNumberUrb	传输数据块数量	IntFeature	√

5.2 功能类定义

5.2.1 Feature

负责查看各种数据类型功能的基础功能：获取功能名称，判断其是否已实现、可读、可写。

Feature 类是 [IntFeature](#)/[FloatFeature](#)/[EnumFeature](#)/[BoolFeature](#)/[StringFeature](#)/[BufferFeature](#)/[CommandFeature](#) 属性类的父类。

5.2.1.1 接口列表

String	getName();	获取功能名称字符串
boolean	isImplemented();	判断属性参数是否已实现
boolean	isReadable();	判断属性参数是否可读
boolean	isWritable ();	判断属性参数是否可写

示例代码：

```
// 获取功能名称
String strName = cam.StreamTransferSize.getName();

// 获取功能是否实现
boolean isImplemented = cam.StreamTransferSize.isImplemented();
if (isImplemented)
{
    // 获取是否可写
    boolean isWritable = cam.StreamTransferSize.isWritable();
    if (isWritable)
    {
        // 设置 URB 大小
        cam.StreamTransferSize.set(512*1024);
    }

    // 获取是否可读
    boolean isReadable = cam.StreamTransferSize.isReadable();
    if (isReadable)
    {
        // 获取 URB 大小当前值
```

```
        long transferSize = cam.StreamTransferSize.get();  
    }  
}
```

5.2.1.2 接口说明

5.2.1.2.1 getName

声明:

String Feature.getName();

意义:

获取功能名称

返回值:

true: 实现

false: 未实现

异常处理:

获取失败, 返回 ""。

5.2.1.2.2 isImplemented

声明:

boolean Feature.isImplemented();

意义:

判断属性参数是否已实现

返回值:

true: 实现

false: 未实现

异常处理:

- 1) 如果属性参数是无效参数, 则返回 false;
- 2) 因为其他原因导致的获取属性参数是否实现失败, 则抛出异常, 异常类型详见 [3.8.2 节](#)。

5.2.1.2.3 isReadable

声明:

boolean Feature.isReadable();

意义:

判断属性参数是否可读

返回值:

true: 可读

false: 不可读

异常处理:

- 1) 如果功能未实现, 则函数返回 false;
- 2) 如果获取属性参数是否可读失败, 则抛出异常, 异常类型详见 [3.8.2 节](#)。

5.2.1.2.4 isWritable

声明:

```
boolean Feature.isWritable();
```

意义:

判断属性参数是否可写

返回值:

true: 可写

false: 不可写

异常处理:

- 1) 如果功能未实现, 则返回 false;
- 2) 如果获取属性参数是否可写失败, 则抛出异常, 异常类型详见 [3.8.2 节](#)。

5.2.2 IntFeature

负责查看、控制相机的整型值功能, 继承自 [Feature](#) 类。

5.2.2.1 接口列表

String	getName();	获取功能名称字符串
boolean	isImplemented();	判断属性是否已实现
boolean	isReadable();	判断属性是否可读
boolean	isWritable ();	判断属性是否可写
long	getMin();	获取参数最小值
long	getMax();	获取参数最大值
long	getInc();	获取参数步长
long	get();	读取参数值
void	set(long value);	设置参数值

示例代码:

```
// 获取图像宽度最大值  
long max = cam.Width.getMax();
```

```
// 设置当前图像宽度为最大值  
cam.Width.set(max);  
  
// 获取当前图像宽度  
long value = cam.Width.get();
```

5.2.2.2 接口说明

5.2.2.2.1 getMin

声明:

```
long IntFeature.getMin();
```

意义:

获取最小值

返回值:

最小值

异常处理:

如果获取不成功, 则抛出异常, 异常类型详见 [3.8.2 节](#)。

5.2.2.2.2 getMax

声明:

```
long IntFeature.getMax();
```

意义:

获取最大值

返回值:

最大值

异常处理:

如果获取不成功, 则抛出异常, 异常类型详见 [3.8.2 节](#)。

5.2.2.2.3 getStep

声明:

```
long IntFeature.getStep();
```

意义:

获取步长值

返回值:

步长值

异常处理：

如果获取不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.2.2.2.4 get

声明：

```
IntFeature.get();
```

意义：

读取整型属性参数值

返回值：

获取的整型值

异常处理：

如果获取该整型属性参数值不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.2.2.2.5 set

声明：

```
void IntFeature.set(long value);
```

意义：

[in] value 设置整型属性参数值

形参：

整型数值

异常处理：

如果设置该整型属性参数不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.2.3 FloatFeature

负责查看、控制相机的浮点型值功能，继承自 [Feature](#) 类。

5.2.3.1 接口列表

String	getName();	获取功能名称字符串
boolean	isImplemented();	判断属性是否已实现
boolean	isReadable();	判断属性是否可读
boolean	isWritable ();	判断属性是否可写
double	getMin();	获取参数最小值
double	getMax();	获取参数最大值
boolean	hasInc();	判断是否有步长

double	getInc();	获取参数步长
String	getUnit();	获取单位
double	get();	读取参数值
void	set(double value);	设置参数值

示例代码：

```
// 获取曝光值可设置最大值
double max = cam.ExposureTime.getMax();

// 设置当前曝光值为 10ms
cam.ExposureTime.set(10000);

// 获取当前曝光值
double curValue = cam.ExposureTime.get();
```

5.2.3.2 接口说明

5.2.3.2.1 getMin

声明：

```
double FloatFeature.getMin();
```

意义：

获取最小值

返回值：

最小值

异常处理：

如果获取不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.2.3.2.2 getMax

声明：

```
double FloatFeature.getMax();
```

意义：

获取最大值

返回值：

最大值

异常处理：

如果获取不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.2.3.2.3 hasInc

声明:

```
boolean FloatFeature.hasInc();
```

意义:

判断是否支持步长值

返回值:

是否支持步长值

异常处理:

如果执行不成功, 则抛出异常, 异常类型详见 [3.8.2 节](#)。

5.2.3.2.4 getInc

声明:

```
double FloatFeature.getInc();
```

意义:

获取步长值

返回值:

步长值

异常处理:

如果获取不成功, 则抛出异常, 异常类型详见 [3.8.2 节](#)。

5.2.3.2.5 getUnit

声明:

```
String FloatFeature.getUnit();
```

意义:

获取单位

返回值:

单位

异常处理:

如果获取不成功, 则抛出异常, 异常类型详见 [3.8.2 节](#)。

5.2.3.2.6 get

声明:

```
double FloatFeature.get();
```

意义:

读取浮点型属性参数值

返回值:

获取的浮点型属性参数值

异常处理:

如果获取浮点型属性参数不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.2.3.2.7 set

声明:

```
void FloatFeature.set(double value);
```

意义:

设置浮点型属性参数值

形参:

[in] value 设置的浮点型数值

异常处理:

如果设置该浮点型属性参数不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.2.4 EnumFeature

负责查看、控制相机的枚举型值功能，继承自 [Feature](#) 类。

5.2.4.1 接口列表

String	getName();	获取功能名称字符串
boolean	isImplemented();	判断属性是否已实现
boolean	isReadable();	判断属性是否可读
boolean	isWritable ();	判断属性是否可写
List<EnumDescription>	getRange();	获取支持的枚举项列表
long	get();	读取参数值
void	set(long value);	设置参数值

枚举量描述类 EnumDescription 中的信息如下:

value	枚举项值
symbolic	枚举项描述

示例代码:

```
// 获取枚举值可设置范围
List<EnumDescription> enumDescList = cam.BalanceWhiteAuto.getRange();
```



```
// 设置当前枚举值 (开启连续自动白平衡)
cam.BalanceWhiteAuto.set (CONTINUOUS.getValue());

// 获取当前自动白平衡状态
long value = cam.BalanceWhiteAuto.get();
```

5.2.4.2 接口说明

5.2.4.2.1 getRange

声明:

```
List<EnumDescription> EnumFeature.getRange();
```

意义:

获取支持的枚举项列表

返回值:

支持的枚举项列表

异常处理:

如果获取该枚举型属性参数范围不成功, 则抛出异常, 异常类型详见 [3.8.2 节](#)。

5.2.4.2.2 get

声明:

```
long EnumFeature.get();
```

意义:

读取枚举型属性参数的值

返回值:

枚举型属性的值

异常处理:

如果获取枚举型属性参数值不成功, 则抛出异常, 异常类型详见 [3.8.2 节](#)。

5.2.4.2.3 set

声明:

```
void EnumFeature.set( long enumValue);
```

意义:

设置枚举型属性参数值

形参:

[in]enumValue 设置的枚举型数值

异常处理：

如果设置该枚举型属性参数不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.2.5 BoolFeature

负责查看、控制相机的布尔型值功能，继承自 [Feature](#) 类。

5.2.5.1 接口列表

String	getName();	获取功能名称字符串
boolean	isImplemented();	判断属性是否已实现
boolean	isReadable();	判断属性是否可读
boolean	isWritable ();	判断属性是否可写
boolean	get();	读取参数值
void	set(boolean value);	设置参数值

示例代码：

```
// 设置当前布尔值
cam.LineInverter.set(true);

// 读取布尔值
boolean value = cam.LineInverter.get();
```

5.2.5.2 接口说明

5.2.5.2.1 get

声明：

```
boolean BoolFeature.get();
```

意义：

读取布尔型属性参数值

返回值：

获取的布尔值

异常处理：

如果获取布尔型属性参数值不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.2.5.2.2 set

声明：

```
void BoolFeature.set(boolean value);
```

意义:

设置浮点型属性参数值

形参:

[in] value 布尔型数值

异常处理:

如果设置该布尔型属性参数不成功, 则抛出异常, 异常类型详见 [3.8.2 节](#)。

5.2.6 StringFeature

负责查看、控制相机的字符串型值功能, 继承自 [Feature](#) 类。

5.2.6.1 接口列表

String	getName();	获取功能名称字符串
boolean	isImplemented();	判断属性是否已实现
boolean	isReadable();	判断属性是否可读
boolean	isWritable ();	判断属性是否可写
int	getMaxLength();	获取字符串可设置的最大长度
String	get();	读取参数值
void	set(String value);	设置参数值

示例代码:

```
// 读取最长字符串长度
int maxLength = cam.DeviceUserID.getMaxLength();

// 读取当前字符串值
String curValue = cam.DeviceUserID.get();

// 设置字符串值
cam.DeviceUserID.set("MyUserID");
```

5.2.6.2 接口说明

5.2.6.2.1 getMaxLength

声明:

int StringFeature.getMaxLength();

意义:

获取字符串型参数可设置的最大长度

返回值:

字符串型参数可设置的最大长度

异常处理:

如果获取字符串型属性参数值可设置最大长度不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.2.6.2.2 get

声明:

```
String StringFeature.get();
```

意义:

读取字符串型属性参数值

返回值:

获取的字符串型属性参数值

异常处理:

如果获取该字符串型属性参数值不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.2.6.2.3 set

声明:

```
void StringFeature.set(String value);
```

意义:

设置字符串型属性参数值

形参:

[in] value 设置的字符串型数值

异常处理:

如果设置该字符串型属性参数不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.2.7 BufferFeature

负责查看、控制相机的 Buffer 类型功能，继承自 [Feature](#) 类。

5.2.7.1 接口列表

String	getName();	获取功能名称字符串
boolean	isImplemented();	判断属性是否已实现
boolean	isReadable();	判断属性是否可读
boolean	isWritable ();	判断属性是否可写
int	getLength();	获取 Buffer 型属性参数的长度

ByteBuffer	get();	读取参数值
void	set(ByteBuffer value);	设置参数值

示例代码：

```
// 读取缓冲数据长度
int bufferLength = cam.LUTValueAll.getLength();

// 读取块数据 (LUT)
ByteBuffer bufferData = cam.LUTValueAll.get();

// 设置块数据 (LUT)
cam.LUTValueAll.set(bufferData);
```

5.2.7.2 接口说明

5.2.7.2.1 getLength

声明：

```
int BufferFeature.getLength();
```

意义：

获取 Buffer 型属性参数的长度

返回值：

Buffer 型属性参数的长度

异常处理：

如果获取 Buffer 型属性参数长度不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.2.7.2.2 get

声明：

```
ByteBuffer BufferFeature.get();
```

意义：

读取 Buffer 型属性参数数据

返回值：

ByteBuffer 对象

异常处理：

如果获取该 Buffer 型属性参数数据不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.2.7.2.3 set

声明：

```
void BufferFeature.set(ByteBuffer value);
```

意义：

设置 Buffer 型属性参数数据

形参：

[in] value 设置 Buffer 类型参数值

异常处理：

如果设置该 Buffer 型参数不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.2.8 CommandFeature

负责相机的命令型功能，继承自 [Feature](#) 类。

5.2.8.1 接口列表

String	getName();	获取功能名称字符串
boolean	isImplemented();	判断属性是否已实现
boolean	isReadable();	判断属性是否可读
boolean	isWritable ();	判断属性是否可写
void	sendCommand();	发送命令

示例代码：

```
// 发送软触发命令
cam.TriggerSoftware.sendCommand();
```

5.2.8.2 接口说明

5.2.8.2.1 sendCommand

声明：

void CommandFeature.sendCommand();

意义：

发送命令

异常处理：

如果发送命令不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.2.9 StaticDefectCorrection

静态坏点矫正数据结构。

5.2.9.1 接口列表

int	width;	图像坏点矫正区域宽度
int	height;	图像坏点矫正区域高度
int	offsetX;	图像坏点矫正区域起点 X 坐标
int	offsetY;	图像坏点矫正区域起点 Y 坐标
int	widthMax;	图像坏点矫正区域最大值（图像最大宽度）
EnumDefineSet.PixelColorFilter	bayerType;	图像数据格式，见 PixelColorFilterEntry
EnumDefineSet.PixelSizeEntry	actualBits;	图像数据有效位，见 PixelSizeEntry

示例代码：

```
// 创建图像处理对象

int nWidth = 5120;
int nHeight = 5120;
int nOffsetX = 0;
int nOffsetY = 0;
int nWidthMax = 5120

GxIJNI.StaticDefectCorrection StaticPa = new GxIJNI.StaticDefectCorrection(
    nWidth, nHeight,
    nOffsetX, nOffsetY, nWidthMax,
    EnumDefineSet.PixelColorFilter.GB,
    EnumDefineSet.PixelSizeEntry.BPP8);

ByteBuffer byInputBuffer = new
    ByteBuffer("/sdcard/Download/w5120_h5120_bayerGB8.raw");

ByteBuffer byDbBuffer = new
    ByteBuffer("/sdcard/Download/w5120_h5120_bayerGB8.dp");

//Define output buffer
ByteBuffer byOutBuffer = new ByteBuffer(byInputBuffer.getLength());

imageProcess.staticDefectCorrection(byInputBuffer.getData(),
    byOutBuffer.getData(),
    StaticPa,
    byDbBuffer.getData(),
    byDbBuffer.getLength());
```

5.3 数据类型定义

5.3.1 DeviceClass

定义	值	解释
UNKNOWN	0	未知设备种类
USB2	1	USB2.0 相机
GEV	2	GEV 相机 (Gige Vision);
U3V	3	USB3.0 相机 (USB3 Vision);

5.3.2 AccessStatus

定义	值	解释
UNKNOWN	0	设备当前状态未知
READWRITE	1	设备当前可读可写
READONLY	2	设备当前仅支持读
NOACCESS	3	设备当前既不支持读，又不支持写

5.3.3 AccessMode

定义	值	解释
READONLY	2	以只读方式打开设备
CONTROL	3	以控制方式打开设备
EXCLUSIVE	4	以独占方式打开设备

5.3.4 FrameStatus

定义	值	解释
SUCCESS	0	正常帧
IMCOMPLETE	-1	残帧
INVALID_IMAGE_INFO	-2	图像头信息无效

5.3.5 PixelFormatEntry

定义	值	解释
UNDEFINED	0x00000000	未定义
MONO8	0x01080001	Monochrome 8-bit
MONO8_SIGNED	0x01080002	Monochrome 8-bit signed
MONO10	0x01100003	Monochrome 10-bit unpacked
MONO12	0x01100005	Monochrome 12-bit unpacked

MONO14	0x01100025	Monochrome 14-bit unpacked
MONO16	0x01100007	Monochrome 16-bit
BAYER_GR8	0x01080008	Bayer Green-Red 8-bit
BAYER_RG8	0x01080009	Bayer Red-Green 8-bit
BAYER_GB8	0x0108000A	Bayer Green-Blue 8-bit
BAYER_BG8	0x0108000B	Bayer Blue-Green 8-bit
BAYER_GR10	0x0110000C	Bayer Green-Red 10-bit
BAYER_RG10	0x0110000D	Bayer Red-Green 10-bit
BAYER_GB10	0x0110000E	Bayer Green-Blue 10-bit
BAYER_BG10	0x0110000F	Bayer Blue-Green 10-bit
BAYER_GR12	0x01100010	Bayer Green-Red 12-bit
BAYER_RG12	0x01100011	Bayer Red-Green 12-bit
BAYER_GB12	0x01100012	Bayer Green-Blue 12-bit
BAYER_BG12	0x01100013	Bayer Blue-Green 12-bit
BAYER_GR16	0x0110002E	Bayer Green-Red 16-bit
BAYER_RG16	0x0110002F	Bayer Red-Green 16-bit
BAYER_GB16	0x01100030	Bayer Green-Blue 16-bit
BAYER_BG16	0x01100031	Bayer Blue-Green 16-bit
RGB8_PLANAR	0x02180021	Red-Green-Blue 8-bit planar
RGB10_PLANAR	0x02300022	Red-Green-Blue 10-bit planar
RGB12_PLANAR	0x02300023	Red-Green-Blue 12-bit planar
RGB16_PLANAR	0x02300024	Red-Green-Blue 16-bit planar

5.3.6 PixelSizeEntry

定义	值	解释
BPP8	8	像素大小 BPP8
BPP10	10	像素大小 BPP10
BPP12	12	像素大小 BPP12
BPP16	16	像素大小 BPP16
BPP24	24	像素大小 BPP24
BPP30	30	像素大小 BPP30
BPP32	32	像素大小 BPP32
BPP36	36	像素大小 BPP36
BPP48	48	像素大小 BPP48
BPP64	64	像素大小 BPP64

5.3.7 PixelColorFilterEntry

定义	值	解释
NONE	0	无
BAYER_RG	1	RG 格式
BAYER_GB	2	GB 格式

BAYER_GR	3	GR 格式
BAYER_BG	4	BG 格式

5.3.8 AcquisitionModeEntry

定义	值	解释
SINGLE_FRAME	0	单帧模式
MULITI_FRAME	1	多帧模式
CONTINUOUS	2	连续模式

5.3.9 TriggerSourceEntry

定义	值	解释
SOFTWARE	0	软触发
LINE0	1	触发源 0
LINE1	2	触发源 1
LINE2	3	触发源 2
LINE3	4	触发源 3

5.3.10 TriggerActivationEntry

定义	值	解释
FALLING_EDGE	0	下降沿触发
RISING_EDGE	1	上升沿触发

5.3.11 ExposureModeEntry

定义	值	解释
TIMED	1	曝光时间寄存器控制曝光时间
TRIGGER_WIDTH	2	触发信号宽度控制曝光时间

5.3.12 UserOutputSelectorEntry

定义	值	解释
OUTPUT0	1	输出 0
OUTPUT1	2	输出 1
OUTPUT2	4	输出 2

5.3.13 UserOutputModeEntry

定义	值	解释
----	---	----

STROBE	0	闪光灯
USER_DEFINED	1	用户自定义

5.3.14 GainSelectorEntry

定义	值	解释
ALL	0	所有增益通道
RED	1	红通道增益
GREEN	2	绿通道增益
BLUE	3	蓝通道增益

5.3.15 BlackLevelSelectEntry

定义	值	解释
ALL	0	所有黑电平通道
RED	1	红通道黑电平
GREEN	2	绿通道黑电平
BLUE	3	蓝通道黑电平

5.3.16 BalanceRatioSelectorEntry

定义	值	解释
RED	0	红通道
GREEN	1	绿通道
BLUE	2	蓝通道

5.3.17 AALightEnvironmentEntry

定义	值	解释
NATURE_LIGHT	0	自然光
AC50HZ	1	50 赫兹日光灯
AC60HZ	2	60 赫兹日光灯

5.3.18 UserSetEntry

定义	值	解释
DEFAULT	0	默认参数组
USER_SET0	1	用户参数组 0

5.3.19 AWBLampHouseEntry

定义	值	解释
ADAPTIVE	0	自适应光源
D65	1	指定色温 6500k
FLUORESCENCE	2	指定荧光灯
INCANDESCENT	3	指定白炽灯
D75	4	指定色温 7500k
D50	5	指定色温 5000k
U30	6	指定色温 3000k

5.3.20 TestPatternEntry

定义	值	解释
OFF	0	关闭
GRAY_FRAME_RAMP_MOVING	1	静止灰度递增
SLANT_LINE_MOVING	2	滚动斜条纹
VERTICAL_LINE_MOVING	3	滚动竖条纹
HORIZONTAL_LINE_MOVING	4	滚动横条纹
GREY_VERTICAL_RAMP	5	灰度竖条纹
SLANT_LINE	6	静止斜条纹

5.3.21 TriggerSelectorEntry

定义	值	解释
FRAME_START	1	采集一帧
FRAME_BURST_START	2	帧高速连拍开始

5.3.22 LineSelectorEntry

定义	值	解释
LINE0	0	引脚 0
LINE1	1	引脚 1
LINE2	2	引脚 2
LINE3	3	引脚 3
LINE4	4	引脚 4
LINE5	5	引脚 5

5.3.23 LineModeEntry

定义	值	解释
INPUT	0	输入
OUTPUT	1	输出

5.3.24 LineSourceEntry

定义	值	解释
OFF	0	关闭
STROBE	1	闪光灯
USER_OUTPUT0	2	用户自定义输出 0
USER_OUTPUT1	3	用户自定义输出 1
USER_OUTPUT2	4	用户自定义输出 2
EXPOSURE_ACTIVE	5	曝光有效
FRAME_TRIGGER_WAIT	6	单帧触发等待
ACQUISITION_TRIGGER_WAIT	7	多帧触发等待

5.3.25 LutSelectorEntry

定义	值	解释
LUMINANCE	0	亮度

5.3.26 TransferControlModeEntry

定义	值	解释
BASIC	0	基础模式
USER_CONTROLLED	1	用户控制模式

5.3.27 TransferOperationModeEntry

定义	值	解释
MULTI_BLOCK	0	指定发送帧数

5.3.28 TestPatternGeneratorSelectorEntry

定义	值	解释
SENSOR	0	sensor 的测试图
REGION0	1	FPGA 的测试图

5.3.29 ChunkSelectorEntry

定义	值	解释
----	---	----

FRAME_ID	1	帧号
TIME_STAMP	2	时间戳

5.3.30 BinningHorizontalModeEntry

定义	值	解释
SUM	0	BINNING 水平值和
AVERAGE	1	BINNING 水平值平均值

5.3.31 BinningVerticalModeEntry

定义	值	解释
SUM	0	BINNING 垂直值和
AVERAGE	1	BINNING 垂直值平均值

5.3.32 SensorShutterModeEntry

定义	值	解释
GLOBAL	0	全局快门，所有像素同时曝光，且曝光时间相等
ROLLING	1	卷帘曝光，所有像素曝光时间相等，但曝光起始时间不同
GLOBALRESET	2	GlobalReset，所有像素曝光起始时间相同，但曝光时间不相等

5.3.33 AcquisitionStatusSelectorEntry

定义	值	解释
ACQUISITION_TRIGGER_WAIT	0	采集触发等待
FRAME_TRIGGER_WAIT	1	帧触发等待

5.3.34 GammaModeEntry

定义	值	解释
SRGB	0	默认 Gamma 校正
USER	1	用户自定义 Gamma 校正

5.3.35 ColorTransformationModeEntry

定义	值	解释
RGB_TO_RGB	0	默认颜色校正

USER	1	用户自定义颜色校正
------	---	-----------

5.3.36 ColorTransformationValueSelectorEntry

定义	值	解释
GAIN00	0	颜色转换分量增益值 GAIN00
GAIN01	1	颜色转换分量增益值 GAIN01
GAIN02	2	颜色转换分量增益值 GAIN02
GAIN10	3	颜色转换分量增益值 GAIN10
GAIN11	4	颜色转换分量增益值 GAIN11
GAIN12	5	颜色转换分量增益值 GAIN12
GAIN20	6	颜色转换分量增益值 GAIN20
GAIN21	7	颜色转换分量增益值 GAIN21
GAIN22	8	颜色转换分量增益值 GAIN22

5.3.37 AutoEntry

定义	值	解释
OFF	0	关闭
CONTINUOUS	1	连续
ONCE	2	单次

5.3.38 SwitchEntry

定义	值	解释
OFF	0	关闭
ON	1	开启

5.3.39 RegionSendModeEntry

定义	值	解释
SINGLE_ROI	0	单 ROI
MULTI_ROI	1	多 ROI

5.3.40 RegionSelectorEntry

定义	值	解释
REGION0	0	区域 0
REGION1	1	区域 1
REGION2	2	区域 2

REGION3	3	区域 3
REGION4	4	区域 4
REGION5	5	区域 5
REGION6	6	区域 6
REGION7	7	区域 7

5.3.41 BayerConvertType

定义	值	解释
NEIGHBOUR	0	邻域平均插值算法
ADAPTIVE	1	边缘自适应插值算法
NEIGHBOUR3	2	更大区域的邻域平均插值算法

5.3.42 ValidBit

定义	值	解释
BIT0_7	0	0-7 位
BIT1_8	1	1-8 位
BIT2_9	2	2-9 位
BIT3_10	3	3-10 位
BIT4_11	4	4-11 位

5.3.43 ImageMirrorMode

定义	值	解释
HORIZONTAL_MIRROR	0	水平镜像
VERTICAL_MIRROR	1	垂直镜像

5.3.44 ActualBits

定义	值	解释
BITS_10	10	10 位
BITS_12	12	12 位
BITS_14	14	14 位
BITS_16	16	16 位

5.3.45 TimerSelectorEntry

定义	值	解释
TIMER1	1	定时器 1

5.3.46 TimerTriggerSourceEntry

定义	值	解释
EXPOSURE_START	1	曝光开始信号

5.3.47 CounterSelectorEntry

定义	值	解释
COUNTER1	1	计数器 1

5.3.48 CounterEventSourceEntry

定义	值	解释
FRAME_START	1	帧开始

5.3.49 CounterResetSourceEntry

定义	值	解释
OFF	0	无复位源
SOFTWARE	1	软触发
LINE0	2	引脚 0
LINE1	3	引脚 1
LINE2	4	引脚 2
LINE3	5	引脚 3

5.3.50 CounterResetActivationEntry

定义	值	解释
RISING_EDGE	1	上升沿触发

5.4 接口定义

5.4.1 DeviceManager

负责相机设备的管理，包括枚举设备、获取设备列表、打开设备等。

5.4.1.1 接口列表

List<DeviceInfo> updateDeviceList (int timeout);	枚举同一网段中的设备(普通枚举)
List<DeviceInfo> updateAllDeviceList (int timeout);	枚举所有的设备(普通枚举)
void setOnUpdateDeviceListFinishedListener(OnUpdateDeviceListFinishedListener listener);	设置枚举完成监听者（无 Root 权限枚举）

void	startUpdateDeviceList(Context context, int timeout);	开始枚举过程（无 Root 权限枚举）
void	startUpdateAllDeviceList(Context context, int timeout);	开始枚举过程（无 Root 权限枚举），在 U3V 设备与千兆网设备同时使用，而且需 要枚举所有千兆网设备才使用该接口
List<DeviceInfo>	getDeviceInfoList ();	获取已枚举到的设备列表
Device	openDeviceBySN (String strSN, AccessMode accessMode);	通过序列号打开设备
Device	openDeviceByUserID (String strUserID, AccessMode accessMode);	通过用户 ID 号打开设备
Device	openDeviceByIndex (int index, AccessMode accessMode);	通过设备索引打开设备
Device	openDeviceByIP (String strIP, AccessMode accessMode);	通过 IP 地址打开设备
Device	openDeviceByMAC (String strMAC, AccessMode accessMode);	通过 MAC 地址打开设备

示例代码：

参考[打开设备](#)。

5.4.1.2 接口说明

5.4.1.2.1 updateDeviceList

声明：

```
List<DeviceInfo> DeviceManager.updateDeviceList (int timeout);
```

意义：

对于非 GEV 相机，枚举所有设备；对于 GEV 相机，枚举同一网段设备

形参：

[in] timeout 枚举超时[0, 0x7fffffff], 单位：ms

返回值：

设备信息列表 List<DeviceInfo>, DeviceInfo 包含的信息详见[枚举设备](#)

异常处理：

如果获取所有设备信息不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.4.1.2.2 updateAllDeviceList

声明：

```
List<DeviceInfo> DeviceManager.updateAllDeviceList (int timeout);
```

意义：

对于非 GEV 相机，枚举所有设备；对于 GEV 相机，枚举全网设备

形参：

[in] timeout 枚举超时[0, 0x7fffffff], 单位：ms

返回值：

设备信息列表 List<DeviceInfo>, DeviceInfo 包含的信息详见[枚举设备](#)

异常处理：

如果获取所有设备信息不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.4.1.2.3 setOnUpdateDeviceListFinishedListener

声明：

```
void DeviceManager.setOnUpdateDeviceListFinishedListener(  
    OnUpdateDeviceListFinishedListener listener);
```

意义：

设置枚举完成监听者，主要配合 startUpdateDeviceList 或 startUpdateDeviceList 接口使用

形参：

[in] listener 枚举完成监听者

异常处理：

如果设置枚举完成监听者不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.4.1.2.4 startUpdateDeviceList

声明：

```
void DeviceManager.startUpdateDeviceList (Context context, int timeout);
```

意义：

开始枚举过程（无 Root 权限枚举），如果不使用 U3V 相机，**不能使用**本接口；在 U3V 和 GEV 相机同时使用，只能枚举到与主机同一网段内的相机

注意：

- 1) 调用本接口之前，**必须**先调用 setOnUpdateDeviceListFinishedListener 接口设置枚举完成监听者，具体示例代码见 [1.4.2 节](#)
- 2) 不支持多线程同时调用

形参：

[in] context 上下文对象，在 MainActivity 中可以传入 this，也可以通过 getApplicationContext()接口获取 context 传入

[in] timeout 枚举超时[0, 0x7fffffff], 单位：ms

异常处理：

如果开启枚举过程不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.4.1.2.5 startUpdateAllDeviceList

声明：

```
void DeviceManager.startUpdateAllDeviceList (Context context, int timeout);
```

意义：

开始枚举过程（无 Root 权限枚举），如果不使用 U3V 相机，**不能使用**本接口；对于非 GEV 相机，枚举所有设备；对于 GEV 相机，枚举全网设备

注意：

- 1) 调用本接口之前，**必须**先调用 setOnUpdateDeviceListFinishedListener 接口设置枚举完成监听者
- 2) 不支持多线程同时调用

形参：

[in] context 上下文对象，可以通过 getApplicationContext()接口获取

[in] timeout 枚举超时[0, 0x7fffffff]，单位：ms

异常处理：

如果开启枚举过程不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.4.1.2.6 getDeviceInfoList

声明：

```
DeviceManager.getDeviceInfoList();
```

意义：

获取设备信息列表

返回值：

设备信息列表。设备信息列表的元素个数为枚举到的设备个数，列表中元素的为 DeviceInfo，见[枚举设备](#)

5.4.1.2.7 openDeviceBySN

声明：

```
Device DeviceManager.openDeviceBySN (String strSN, AccessMode accessMode);
```

意义：

通过序列号打开设备

形参：

[in] strSN 序列号[字符串类型]

[in] accessMode 打开设备模式，仅对 GEV 相机有效，查看 [AccessMode](#)

返回值：

设备对象

异常处理:

如果打开设备不成功, 则抛出异常, 异常类型详见 [3.8.2 节](#)。

5.4.1.2.8 openDeviceByUserID

声明:

Device DeviceManager.openDeviceByUserID (String strUserID, AccessMode accessMode);

意义:

通过用户 ID 号打开设备

形参:

[in] UserID 用户 ID 号[字符串类型]

[in] accessMode 打开设备访问模式, 仅对 GEV 相机有效, 查看 [AccessMode](#)

返回值:

设备对象

异常处理:

如果打开设备不成功, 则抛出异常, 异常类型详见 [3.8.2 节](#)。

5.4.1.2.9 openDeviceByIndex

声明:

Device DeviceManager.openDeviceByIndex (int index, AccessMode accessMode);

意义:

通过设备索引打开设备

注意:

设备索引从 1 开始

形参:

[in] index 设备索引[1,2,3...设备个数]

[in] accessMode 打开设备方式, 仅对 GEV 相机有效, 查看 [AccessMode](#)

返回值:

设备对象

异常处理:

如果打开设备不成功, 则抛出异常, 异常类型详见 [3.8.2 节](#)。

5.4.1.2.10 openDeviceByIP

声明:

Device DeviceManager.openDeviceByIP (String strIP, AccessMode accessMode);

意义：

通过设备 ip 地址打开 GEV 相机设备

形参：

[in] strIP 设备 ip 地址[字符串类型]

[in] accessMode 打开设备模式，仅对 GEV 相机有效，查看 [AccessMode](#)

返回值：

设备对象

异常处理：

如果打开设备不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.4.1.2.11 openDeviceByMAC

声明：

Device DeviceManager.openDeviceByMAC (String strMAC, AccessMode accessMode);

意义：

通过设备 mac 地址打开 GEV 相机设备

形参：

[in] strMAC 设备 MAC 地址[字符串类型]

[in] accessMode 打开设备模式，仅对 GEV 相机有效，查看 [AccessMode](#)

返回值：

设备对象

异常处理：

如果打开设备不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.4.2 Device

负责相机设备的采集控制、设备关闭、配置文件导入导出和获取设备句柄等。

5.4.2.1 接口列表

int	getStreamChannelNum();	获取当前设备支持的流通道个数
DataStream	getDataStream(int index)	获取 DataStream 对象
void	streamOn ();	开采，相机开始传送图像数据
void	streamOff ();	停采，相机停止传送图像数据
void	exportConfigFile (String filePath);	导出当前配置文件

void	importConfigFile(String filePath, Boolean verify);	导入配置文件
void	closeDevice ();	关闭设备

示例代码：

参考[打开设备](#)、[采集控制](#)。

5.4.2.2 接口说明

5.4.2.2.1 getStreamChannelNum

声明：

```
int Device.getStreamChannelNum();
```

意义：

获取当前设备支持的流通道个数

返回值：

流通道个数

注：目前 GEV 相机、USB3.0、USB2.0 相机均不支持多流通道

5.4.2.2.2 getDataStream

声明：

```
DataStream Device.getDataStream(int index);
```

意义：

获取当前设备支持的流通道个数

形参：

[in] index 流的序号，从 0 开始，暂时支持 0

返回值：

[DataStream](#) 对象

5.4.2.2.3 streamOn

声明：

```
void Device.streamOn();
```

意义：

开始采集，相机开始采集并传送图像数据

异常处理：

如果开始采集不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.4.2.2.4 streamOff

声明:

```
void Device.streamOff();
```

意义:

发送停止命令, 相机停止传送图像数据

异常处理:

如果发送停止命令不成功, 则抛出异常, 异常类型详见 [3.8.2 节](#)。

5.4.2.2.5 exportConfigFile

声明:

```
void Device.exportConfigFile(String filePath);
```

意义:

导出当前配置文件

形参:

[in] filePath 文件路径

异常处理:

如果导出当前配置文件不成功, 则抛出异常, 异常类型详见 [3.8.2 节](#)。

5.4.2.2.6 importConfigFile

声明:

```
void Device.importConfigFile(String filePath, boolean verify);
```

意义:

导入配置文件

形参:

[in] filePath 文件路径

[in] verify 是否所有导入值将被验证一致性

异常处理:

如果导入配置文件不成功, 则抛出异常, 异常类型详见 [3.8.2 节](#)。

5.4.2.2.7 closeDevice

声明:

```
void Device.closeDevice();
```

意义:

关闭设备

注意:

当执行关闭设备后，如果还想使用此相机，请重新打开后再操作

异常处理：

如果关闭设备不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.4.3 DataStream

负责采集图像、采集参数设置以及采集图像统计数据的获取。

5.4.3.1 接口列表

void	setAcquisitionBufferNumber (int bufferSize);	设置采集图像缓存 Buffer 的个数
int	getAcquisitionBufferNumber();	获取采集图像缓存 Buffer 的个数
RawImage	getRawImage (int timeout);	采集一幅 Raw 图像数据
Bitmap	getBitmap(ProcessParam processParam, FrameInfo frameInfo, int timeout);	采集一幅图像并转换成 Bitmap 返回， 图像信息将通过 FrameInfo 类型参数 返回
void	getImageBySurface (Surface surface, ProcessParam processParam, FrameInfo frameInfo, String path, int timeout);	采集一幅图像并转换处理后显示在 SurfaceView 控件上，图像信息将通 过 FrameInfo 类型参数返回
void	flushQueue ();	清除设备库图像缓冲队列

说明：getRawImage/getBitmap/getImageBySurface 三个采集图像的接口，一个应用程序中选择一种使用，**不能同时使用**。

示例代码：

参考[采集控制](#)。

5.4.3.2 接口说明

5.4.3.2.1 setAcquisitionBufferNumber

声明：

```
void DataStream.setAcquisitionBufferNumber(int bufferSize);
```

意义：

设置采集 Buffer 的个数

形参：

[in] bufferSize 采集 Buffer 的个数，范围：[1, 0x7ffffff]

异常处理:

如果设置采集 Buffer 个数不成功, 则抛出异常, 异常类型详见 [3.8.2 节](#)。

5.4.3.2.2 getAcquisitionBufferNumber

声明:

```
int DataStream.getAcquisitionBufferNumber();
```

意义:

获取采集图像缓存 Buffer 个数

形参:

无

返回值:

采集图像缓存 Buffer 个数

异常处理:

如果获取采集图像缓存 Buffer 个数不成功, 则抛出异常, 异常类型详见 [3.8.2 节](#)。

5.4.3.2.3 getRawImage

声明:

```
RawImage DataStream.getRawImage(int timeout);
```

意义:

采集一幅 Raw 图像数据

形参:

[in]timeout 获取超时[0, 0x7fffffff]

返回值:

RawImage 对象

异常处理:

- 1) 如果未采集到图像, 则抛出超时异常;
- 2) 如果接口执行失败, 则抛出其他异常, 异常类型详见 [3.8.2 节](#)。

5.4.3.2.4 getBitmap

声明:

```
Bitmap DataStream.getBitmap(ProcessParam processParam,  
                             FrameInfo frameInfo,  
                             int timeout);
```

意义:

采集一幅图像并转成 Bitmap

形参:

[in] processParam	ProcessParam 类对象，包含图像转换和图像质量提升的相关参数
[in] frameInfo	图像帧信息，具体见 3.5.2 节
[in] timeout	获取超时[0, 0x7fffffff]

返回值:

Bitmap 对象

异常处理:

- 1) 如果采集到的图像为残帧，则返回 null;
- 2) 如果未采集到图像，则抛出超时异常;
- 3) 如果接口执行失败，则抛出其他异常，异常类型详见 [3.8.2 节](#)。

5.4.3.2.5 getImageBySurface

声明:

```
void DataStream.getImageBySurface(Surface surface,  
                                   ProcessParam processParam,  
                                   FrameInfo frameInfo,  
                                   String path,  
                                   int timeout);
```

意义:

采集一幅图像并转换处理后显示在 SurfaceView 控件上，图像信息将通过 FrameInfo 类型参数返回

形参:

[in] Surface	参数为 Surface 类型，传入显示图像的 SurfaceView 控件对应的 Surface 对象
[in] processParam	参数为 ProcessParam 类型，包含图像转换和图像质量提升的相关参数
[out] frameInfo	图像帧信息，具体见 3.5.3 节
[in] path	保存图像的路径和名称，只能保存采集到的 Raw 格式的图像
[in] timeout	获取超时[0, 0x7fffffff]

返回值:

无

异常处理:

- 1) 如果采集到图像是残帧，图像将不会被处理、显示和保存;
- 2) 如果未采集到图像，则抛出超时异常;
- 3) 如果接口执行失败，则抛出其他异常，异常类型详见 [3.8.2 节](#)。

5.4.3.2.6 flushQueue

声明:

```
void DataStream.flushQueue();
```

意义：

清除相机采集缓冲队列

异常处理：

如果清除相机采集缓冲队列不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.4.4 RawImage

负责 Raw 图像相关操作。

5.4.4.1 接口列表

RGBImage	convertToRGB(BayerConvertType type, ValidBit validBit, boolean flip);	转换成 RGB 格式图像
ARGBImage	convertToARGB(BayerConvertType type, ValidBit validBit, boolean flip, byte alpha);	转换成 ARGB 格式图像
RawImage	mirror(ImageMirrorMode mirrorMode, ValidBit validBit);	镜像功能
byte[]	getData();	获取 raw 图像数据
void	save(String path);	保存 raw 图数据
int	getStatus();	获取 raw 图状态
int	getWidth();	获取 raw 图宽度
int	getHeight();	获取 raw 图高度
int	getOffsetX();	获取 raw 图 OffsetX
int	getOffsetY();	获取 raw 图 OffsetY
int	getPixelFormat();	获取图像像素格式
long	getFrameID();	获取帧 ID
long	getTimestamp();	获取时间戳

示例代码：

参考图像采集和处理 [getRawImage 方法](#)。

5.4.4.2 接口说明

5.4.4.2.1 convertToRGB

声明：

RGBImage RawImage.convertToRGB ([BayerConvertType](#) type,
[ValidBit](#) validBit,

```
boolean flip);
```

意义:

通过插值转换成 RGB 格式图像

形参:

[in] type 转换的插值方法, 参考 [BayerConvertType](#)

[in] validBit 有效位数, 将非 8 位 Raw 格式图像转换为 8 位的有效位选择, 参考 [ValidBit](#)

[in] flip 输出的 RGB 图像是否上下翻转

返回值:

RGBImage 对象

异常处理:

如果转换不成功, 则抛出异常, 异常类型详见 [3.8.2 节](#)。

5.4.4.2.2 convertToARGB

声明:

```
ARGBImage RawImage.convertToARGB (BayerConvertType type,  
                                     ValidBit validBit,  
                                     boolean flip,  
                                     byte alpha);
```

意义:

通过插值转换成 ARGB 格式图像

形参:

[in] type 转换的插值方法, 参考 [BayerConvertType](#)

[in] validBit 有效位数, 将非 8 位 Raw 格式图像转换为 8 位的有效位选择, 参考 [ValidBit](#)

[in] flip 输出的 RGB 图像是否上下翻转

[in] alpha Alpha 通道值, 一般传入 0xFF

返回值:

ARGBImage 对象

异常处理:

如果转换不成功, 则抛出异常, 异常类型详见 [3.8.2 节](#)。

5.4.4.2.3 mirror

声明:

```
RawImage RawImage.mirror(ImageMirrorMode mirrorMode,  
                           ValidBit validBit);;
```

意义:

图像镜像，支持水平镜像和垂直镜像两种模式

注意：

如果非 8 位 Raw 格式图像，会先转换出 8 位 Raw 图，再进行镜像处理

形参：

[in] mirrorMode 镜像模式，参考 [ImageMirrorMode](#)

[in] validBit 有效位数，将非 8 位 Raw 格式图像转换为 8 位的有效位选择，参考 [ValidBit](#)

返回值：

ARGBImage 对象

异常处理：

如果处理不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.4.4.2.4 getData

声明：

```
byte[] RawImage.getData();
```

意义：

获取 raw 数据

返回值：

raw 数据

5.4.4.2.5 save

声明：

```
void RawImage.save( String filePath);
```

意义：

保存 raw 图数据

形参：

[in]filePath 文件路径

返回值：

无

异常处理：

如果保存 raw 图未成功，则抛异常。

5.4.4.2.6 getStatus

声明：

```
int RawImage.getStatus();
```

意义:

获取 raw 图状态

返回值:

获取图像状态, 标识图像是否完整, 数据类型参考 [FrameStatus](#)

5.4.4.2.7 getWidth

声明:

```
int RawImage.getWidth();
```

意义:

获取 raw 图宽度

返回值:

raw 图宽度

5.4.4.2.8 getHeight

声明:

```
int RawImage.getHeight();
```

意义:

获取 raw 图高度

返回值:

raw 图高度

5.4.4.2.9 getOffsetX

声明:

```
int RawImage.getOffsetX();
```

意义:

获取 raw 图 OffsetX

返回值:

raw 图 OffsetX

5.4.4.2.10 getOffsetY

声明:

```
int RawImage.getOffsetY();
```

意义:

获取 raw 图 OffsetY

返回值:

raw 图 OffsetY

5.4.4.2.11 getPixelFormat

声明:

```
int RawImage.getPixelFormat();
```

意义:

获取图像像素格式

返回值:

像素格式

5.4.4.2.12 getFrameID

声明:

```
long RawImage.getFrameID();
```

意义:

获取帧 ID

返回值:

帧 ID

5.4.4.2.13 getTimestamp

声明:

```
long RawImage.getTimestamp();
```

意义:

获取时间戳

返回值:

时间戳

5.4.5 RGBImage

负责 RGB 图像操作。

5.4.5.1 接口列表

void	imageImprovement (long colorCorrectionParam, ByteBuffer gammaLut, ByteBuffer contrastLut);	图像质量提升
byte[]	getData();	获取 RGB 图像数据
ARGBImage	convertToARGB(byte alpha);	转换 ARGB 数据
int	getWidth();	获取图像宽度
int	getHeight();	获取图像高度
int	getOffsetX();	获取图像 OffsetX

int	getOffsetY();	获取图像 OffsetY
long	getFrameID();	获取帧 ID
long	getTimestamp();	获取时间戳

示例代码：

参考图像采集与处理 [从 Raw 图转换 RGB](#)。

5.4.5.2 接口说明

5.4.5.2.1 imageImprovement

声明：

```
void RGBImage. imageImprovement (long colorCorrectionParam,
                                   ByteBuffer gammaLut,
                                   ByteBuffer contrastLut);
```

意义：

图像质量提升

形参：

[in] colorCorrectionParam	颜色校正参数， 必须 从相机中读取
[in] gammaLut	gamma LUT，为 null 时，不进行相关处理
[in] contrastLut	对比度 LUT，为 null 时，不进行相关处理

异常处理：

- 1) 如果 colorCorrectionParam 为 0 ，且 gamma LUT 和 contrastLut 为 null，函数不做任何处理；
- 2) 如果处理出错，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.4.5.2.2 convertToARGB

声明：

```
ARGBImage RGBImage. convertToARGB (byte alpha);
```

意义：

通过插值转换成 ARGB 格式图像

形参：

[in] alpha	Alpha 通道值，一般传入 0xFF
------------	---------------------

返回值：

ARGBImage 对象

异常处理：

如果转换不成功，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.4.5.2.3 getData

声明:

```
byte[] RGBImage.getData();
```

意义:

获取 RGB 图像数据

返回值:

RGB 图像数据

5.4.5.2.4 getStatus

声明:

```
int RGBImage.getStatus();
```

意义:

获取图像状态, 标识图像是否完整

返回值:

图像状态, 数据类型参考 [FrameStatus](#)

5.4.5.2.5 getWidth

声明:

```
int RGBImage.getWidth();
```

意义:

获取图像宽度

返回值:

图像宽度

5.4.5.2.6 getHeight

声明:

```
int RGBImage.getHeight();
```

意义:

获取图像高度

返回值:

图像高度

5.4.5.2.7 getOffsetX

声明:

```
int RGBImage.getOffsetX();
```

意义:

获取图像 OffsetX

返回值:

图像 OffsetX

5.4.5.2.8 getOffsetY

声明:

int RGBImage.getOffsetY();

意义:

获取图像 OffsetY

返回值:

raw 图 OffsetY

5.4.5.2.9 getFrameID

声明:

long RGBImage.getFrameID();

意义:

获取帧 ID

返回值:

帧 ID

5.4.5.2.10 getTimestamp

声明:

long RGBImage.getTimestamp();

意义:

获取时间戳

返回值:

时间戳

5.4.6 ARGBImage

负责 ARGB 图像操作。

5.4.6.1 接口列表

void	imageImprovement (long colorCorrectionParam, ByteBuffer gammaLut, ByteBuffer contrastLut);	图像质量提升
int[]	getData();	获取 ARGB 图像数据
Bitmap	getBitmap();	获取 Bitmap

int	getWidth();	获取图像宽度
int	getHeight();	获取图像高度
int	getOffsetX();	获取图像 OffsetX
int	getOffsetY();	获取图像 OffsetY
long	getFrameID();	获取帧 ID
long	getTimestamp();	获取时间戳

示例代码：

参考图像采集与处理 [从 Raw 图转换 ARGB](#)。

5.4.6.2 接口说明

5.4.6.2.1 imageImprovement

声明：

```
void ARGBImage.imageImprovement (long colorCorrectionParam,
                                   ByteBuffer gammaLut,
                                   ByteBuffer contrastLut);
```

意义：

图像质量提升

形参：

[in] colorCorrectionParam	颜色校正参数， 必须 从相机中读取
[in] gammaLut	gamma LUT，为 null 时，不进行相关处理
[in] contrastLut	对比度 LUT，为 null 时，不进行相关处理

异常处理：

- 1) 如果 colorCorrectionParam 为 0 ， 且 gamma LUT 和 contrastLut 为 null，函数不做任何处理；
- 2) 如果处理出错，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.4.6.2.2 getBitmap

声明：

```
Bitmap ARGBImage.getBitmap()
```

意义：

从 ARGBImage 中获取 Bitmap

返回值：

Bitmap 对象

5.4.6.2.3 getData

声明：

```
int[] ARGBImage.getData();
```

意义:

获取 ARGB 数据

返回值:

ARGB 数据

5.4.6.2.4 getStatus

声明:

```
int ARGBImage.getStatus();
```

意义:

获取图像状态, 标识图像是否完整

返回值:

图像状态, 数据类型参考 [FrameStatus](#)

5.4.6.2.5 getWidth

声明:

```
int ARGBImage.getWidth();
```

意义:

获取图像宽度

返回值:

图像宽度

5.4.6.2.6 getHeight

声明:

```
int ARGBImage.getHeight();
```

意义:

获取图像高度

返回值:

图像高度

5.4.6.2.7 getOffsetX

声明:

```
int ARGBImage.getOffsetX();
```

意义:

获取图像 OffsetX

返回值:

图像 OffsetX

5.4.6.2.8 getOffsetY

声明:

int ARGBImage.getOffsetY();

意义:

获取图像 OffsetY

返回值:

图像 OffsetY

5.4.6.2.9 getFrameID

声明:

long ARGBImage.getFrameID();

意义:

获取帧 ID

返回值:

帧 ID

5.4.6.2.10 getTimestamp

声明:

long ARGBImage.getTimestamp();

意义:

获取时间戳

返回值:

时间戳

5.4.7 ByteBuffer

负责 ByteBuffer 类的相关操作，主要负责 LUT 等数据的存储并传入本接口库；同时封装了从文件中读取数据并存储的操作。

5.4.7.1 接口列表

ByteBuffer();	无参构造函数
ByteBuffer(int length);	传入 Buffer 大小的构造函数
ByteBuffer(byte[] bytes);	传入 byte 数组的构造函数
ByteBuffer(String path);	传入文件路径的构造函数，从文件中读取数据

byte[]	getData();	获取数据
int	getLength();	获取数据长度
void	save(String path);	保存数据到文件

5.4.7.2 接口说明

5.4.7.2.1 getData

声明:

byte[] ByteBuffer.getData();

意义:

返回 ByteBuffer 对象的数据

返回值:

数据(byte 数组)

5.4.7.2.2 getLength

声明:

int ByteBuffer.getLength();

意义:

返回 ByteBuffer 对象的数据长度

返回值:

数据长度

5.4.7.2.3 save

声明:

void ByteBuffer.save(String filePath);

意义:

保存数据

形参:

[in] filePath 文件路径

返回值:

无

异常处理:

如果保存未成功，则抛异常。

5.4.8 Utility

负责参数 gamma 和 contrast 的操作。

5.4.8.1 接口列表

ByteBuffer	getGammaLut(double gamma);	通过 gamma 值获取 gamma 查找表
ByteBuffer	getContrastLut(int contrast);	通过对比度值获取对比度查找表

示例代码：

参考[图像质量提升](#)。

5.4.8.2 接口说明

5.4.8.2.1 getGammaLut

声明：

ByteBuffer Utility. getGammaLut(double gamma)

意义：

通过 gamma 值获取 gamma 查找表

形参：

[in] gamma 整型或浮点型，范围[0.1, 10.0]

返回值：

gamma 查找表的 Buffer 类型对象

异常处理：

如果获取 gamma 查找表失败，则抛出异常，异常类型详见[3.8.2 节](#)。

5.4.8.2.2 getContrastLut

声明：

ByteBuffer Utility. getContrastLut(int contrast);

意义：

通过对比度值获取对比度查找表

形参：

[in] contrast 整型，范围[-50, 100]

返回值：

对比度查找表

异常处理：

如果获取对比度查找表失败，则抛出异常，异常类型详见 [3.8.2 节](#)。

5.4.9 ImageFormatConvert

负责图像格式转换。表格中为当前支持的转换图像格式，源图像格式与目的图像格式均存在一对多关系（即源图像格式中任一格式可转换为对应的目的图像格式的任意一种）。

	源图像格式	目的图像格式
Bayer 8bit	BAYER_GR8, BAYER_RG8 BAYER_GB8, BAYER_BG8	MONO8, RGB8, BGR8, RGBA8, BGRA8, ARGB8, ABGR8, RGB8_PLANAR
Bayer 16bit	BAYER_GR10, BAYER_RG10, BAYER_GB10, BAYER_BG10, BAYER_GR12, BAYER_RG12, BAYER_GB12, BAYER_BG12, BAYER_GR16, BAYER_RG16, BAYER_GB16, BAYER_BG16	MONO16, RGB16, BGR16, RGB8, BGR8, RGB16_PLANAR
RGB 8bit	RGB8, BGR8	MONO8, RGB8, RGBA8, BGRA8, ARGB8, ABGR8
RGB 16bit	RGB16, BGR16	MONO16

5.4.9.1 接口列表

void creatImageFormatConvertHandle()	创建图像格式转换句柄
void destroyImageFormatConvertHandle()	销毁图像格式转化句柄
void setDstFormat(PixelFormatEntry pixelFormat)	设置期望转换格式
PixelFormatEntry getDstFormat()	获取期望转换格式
void setAlphaValue(int alphaValue)	设置带有 Alpha 通道图像的 Alpha 值
int getAlphaValue()	获取 Alpha 通道值
void setValidBits(ValidBit validBits)	设置输入源图像转换为期望的图像格式时保留的有效数据位
ValidBit getValidBits()	获取输入源图像转换为期望的图像格式时保留的有效数据位
void setInterpolationType(BayerConvertType interpolationType)	设置图像格式转换算法
BayerConvertType getInterpolationType()	获取图像格式转换算法
int getBufferSizeForConversion(int width, int height, int pixelFormat)	根据输入图像指针获取期望格式的图像 Buffer 长度

<pre>void convert(byte[] inputBuffer, int inputSize, byte[] outputBuffer, int outputSize, PixelFormatEntry pixelFormat, int width, int height, boolean flip)</pre>	将输入源图像转换为期望的图像格式
<pre>void convert(byte[] inputBuffer, int inputSize, Bitmap outputBitmap, int outputSize, PixelFormatEntry pixelFormat, int width, int height, boolean flip)</pre>	将输入源图像转换为期望的图像格式

5.4.9.2 接口说明

➤ createImageFormatConvertHandle

声明：

```
void ImageFormatConvert.createImageFormatConvertHandle();
```

意义：

创建图像格式转换句柄

形参：

无

返回值：

无

异常处理：

如果创建失败，则抛出异常

➤ destroyImageFormatConvertHandle

声明：

```
void ImageFormatConvert.destroyImageFormatConvertHandle();
```

意义：

销毁图像格式转换句柄

形参：

无

返回值：

无

异常处理:

如果销毁失败, 则抛出异常

➤ **setDstFormat**

声明:

```
void ImageFormatConvert.setDstFormat(PixelFormatEntry pixelFormat);
```

意义:

设置期望转换格式

形参:

[in] pixelFormat 期望的转换格式, 见定义 [PixelFormatEntry](#)

返回值:

无

异常处理:

如果设置失败, 则抛出异常

➤ **getDstFormat**

声明:

```
PixelFormatEntry ImageFormatConvert.getDstFormat();
```

意义:

获取期望转换格式

形参:

无

返回值:

期望的转换格式

异常处理:

如果获取失败, 则抛出异常

➤ **setAlphaValue**

声明:

```
void ImageFormatConvert.setAlphaValue(int alphaValue);
```

意义:

设置带有 Alpha 通道图像的 Alpha 值

形参:

[in] alphaValue Alpha 通道值, 取值范围 0~255

返回值:

无

异常处理:

如果设置失败, 则抛出异常

➤ **getAlphaValue**

声明:

```
int ImageFormatConvert.getAlphaValue();
```

意义:

获取 Alpha 通道值

形参:

无

返回值:

Alpha 通道值

异常处理:

无

➤ **setValidBits**

声明:

```
void ImageFormatConvert.setValidBits(ValidBit validBits);
```

意义:

设置输入源图像转换为期望的图像格式时保留的有效数据位

形参:

[in] validBits 有效数据位, 见定义 [ValidBit](#)

返回值:

无

异常处理:

如果设置失败, 则抛出异常

➤ **getValidBits**

声明:

```
int ImageFormatConvert.getValidBits();
```

意义:

获取输入源图像转换为期望的图像格式时保留的有效数据位

形参:

无

返回值:

有效数据位

异常处理:

无

➤ **setInterpolationType**

声明:

```
void ImageFormatConvert.setInterpolationType(BayerConvertType interpolationType);
```

意义:

设置图像格式转换算法

形参:

[in] interpolationType 转换算法, 见定义 [BayerConvertType](#)

返回值:

无

异常处理:

如果设置失败, 则抛出异常

➤ **getInterpolationType**

声明:

```
BayerConvertType ImageFormatConvert.getInterpolationType();
```

意义:

获取图像格式转换算法

形参:

无

返回值:

转换算法

异常处理:

无

➤ **getBufferSizeForConversion**

声明:

```
int ImageFormatConvert.getBufferSizeForConversion(int width, int height, int pixelFormat);
```

意义:

根据输入图像指针获取期望格式的图像 Buffer 长度

形参:

[in] width 图像宽
[in] height 图像高
[in] pixelFormat 图像格式

返回值:

图像 Buffer 长度

异常处理:

如果获取失败, 则抛出异常

➤ **convert**

声明:

```
void ImageFormatConvert.convert(byte[] inputBuffer,  
                                int inputSize,  
                                byte[] outputBuffer,  
                                int outputSize,  
                                PixelFormatEntry pixelFormat,  
                                int width,  
                                int height,  
                                boolean flip);
```

意义:

将输入源图像转换为期望的图像格式

形参:

[in] inputBuffer 源图像 Buffer 指针
[in] inputSize 源图像 Buffer 长度
[out] outputBuffer 目的图像 Buffer 指针, 该内存由用户申请, 长度为 outputSize
[in] outputSize 目的图像 Buffer 长度, 可通过 `getBufferSizeForConversion()` 获得
[in] pixelFormat 源图像格式, 见定义 [PixelFormatEntry](#)
[in] width 源图像宽度
[in] height 源图像高度
[in] flip 图像是否翻转 (true-翻转, false-不翻转)

返回值:

无

异常处理:

如果转换失败, 则抛出异常

5.4.9.2.1 convert (转 Bitmap)

声明:

```
void ImageFormatConvert.convert(byte[] inputBuffer,
                                int inputSize,
                                Bitmap outputBitmap,
                                int outputSize,
                                PixelFormatEntry pixelFormat,
                                int width,
                                int height,
                                boolean flip);
```

意义：

将输入源图像转换为期望的图像格式

形参：

- [in]

inputBuffer

源图像 Buffer 指针
- [in]

inputSize

源图像 Buffer 长度
- [out]

outputBitmap

目的图像 Bitmap，该对象由用户申请
- [in]

outputSize

目的图像 Buffer 长度，可通过 `getBufferSizeForConversion()`获得
- [in]

pixelFormat

源图像格式，见定义 [PixelFormatEntry](#)
- [in]

width

源图像宽度
- [in]

height

源图像高度
- [in]

flip

图像是否翻转（true-翻转，false-不翻转）

返回值：

无

异常处理：

如果转换失败，则抛出异常

5.4.10 ImageProcess

负责图像处理功能，可提升图像质量，包括静态坏点校正、动态坏点校正、亮度对比度调节、系统参数设置等功能。

5.4.10.1 接口列表

<div>void staticDefectCorrection(byte[] inputBuffer,</div> <div>byte[] outputBuffer,</div> <div>StaticDefectCorrection defectCorrection,</div> <div>byte[] defectBuffer,</div> <div>int defectBufferSize)</div>	静态坏点校正
<div>void autoRawDefectivePixelCorrect(byte[] inputBuffer,</div> <div>int width,</div> <div>int height,</div>	动态坏点校正

int bitNum, int mode)	
void sharpen24B(byte[] inputBuffer, byte[] outputBuffer, int width, int height, float factor)	锐化
void calcCameraLutBuffer(int contrastParam, double gamma, int lightness, byte[] lutBuffer, int[] lutLength)	计算相机查找表
void setSystem(int systemParam, int value)	设置系统参数
int getSystem(int systemParam)	获取系统参数
void readLutFile(byte[] lutFilePath, byte[] lutBuffer, int[] lutBufferLength)	从文件获取读取查找表数据

5.4.10.2 接口说明

➤ staticDefectCorrection

声明：

```
void ImageProcess.staticDefectCorrection(byte[] inputBuffer,
byte[] outputBuffer,
StaticDefectCorrection defectCorrection,
byte[] defectBuffer,
int defectBufferSize);
```

意义：

对 Raw 图进行静态坏点校正

形参：

[in] inputBuffer	源图像 Buffer 指针
[out] outputBuffer	目的图像 Buffer 指针
[in] defectCorrection	源图像参数，详见 StaticDefectCorrection
[in] defectBuffer	坏点位置数据（从坏点文件中读取数据，坏点文件通过静态坏点校正插件保存的 dp 文件）
[in] defectBufferSize	坏点位置数据 Buffer 大小

返回值：

无

异常处理：

如果校正失败，则抛出异常

➤ **autoRawDefectivePixelCorrect**

声明：

```
void ImageProcess.autoRawDefectivePixelCorrect(byte[] inputBuffer,
                                              int width,
                                              int height,
                                              int bitNum,
                                              int mode);
```

意义：

对 Raw 图进行动态坏点校正

形参：

[in&out] inputBuffer	Raw 图输入/输出 buffer 指针
[in] width	图像宽
[in] height	图像高
[in] bitNum	图像数据的实际位数（注：若是 10 位数据则输入 10，12 位数据输入 12，以此类推，值范围 8 ~ 16）
[in] mode	坏点校正模式，预留

返回值：

无

异常处理：

如果校正失败，则抛出异常

➤ **sharpen24B**

声明：

```
void ImageProcess.sharpen24B(byte[] inputBuffer,
                             byte[] outputBuffer,
                             int width,
                             int height,
                             float factor);
```

意义：

对 RGB 图进行锐化

形参：

[in] inputBuffer	源图像 Buffer 指针
[out] outputBuffer	目的图像 Buffer 指针
[in] width	图像宽度
[in] height	图像高度
[in] factor	调节因子，范围：0.1 ~ 5.0

返回值:

无

异常处理:

如果处理失败, 则抛出异常

➤ **calcCameraLutBuffer**

声明:

```
void ImageProcess.calcCameraLutBuffer(int contrastParam,  
                                     double gamma,  
                                     int lightness,  
                                     byte[] lutBuffer,  
                                     int[] lutLength);
```

意义:

计算相机查找表

形参:

[in] contrastParam	对比度, 取值范围: -50 ~ 100
[in] gamma	Gamma, 取值范围: 0.1 ~ 10
[in] lightness	亮度, 取值范围: -150 ~ 150
[out] lutBuffer	查找表内存指针, 该内存由用户申请, 长度为 lutLength
[in] lutLength	查找表长度, 可通过 device.getBufferFeature("LUTValueAll").getLength()获取

返回值:

无

异常处理:

如果计算失败, 则抛出异常

➤ **setSystem**

声明:

```
void ImageProcess.setSystem(int systemParam, int value);
```

意义:

设置系统参数, 比如多线程数量、SSSE3 使能状态、NEON 使能状态等

形参:

[in] systemParam	系统参数
[in] value	系统参数取值, 线程数量>0; NEON 使能取值 1, 不使能取值 0

返回值:

无

异常处理：

如果设置失败，则抛出异常

➤ **getSystem**

声明：

```
int ImageProcess.getSystem(int systemParam);
```

意义：

获取系统参数

形参：

[in] systemParam 系统参数

返回值：

系统参数取值

异常处理：

如果获取失败，则抛出异常

➤ **readLutFile**

声明：

```
void ImageProcess.readLutFile(byte[] lutFilePath, byte[] lutBuffer, int[] lutBufferLength);
```

意义：

从查坐标文件里读取查坐标的内容

形参：

[in] lutFilePath 查找表文件路径

[out] lutBuffer 查找表内存指针，该内存由用户申请，长度为 lutBufferLength

[in] lutBufferLength 查找表有效数据长度

返回值：

无

异常处理：

如果获取失败，则抛出异常

6 版本说明

序号	修订版本	所做改动	发布日期
1	V1.0.0	初始发布	2019-05-14
2	V1.0.1	添加 SensorShutterMode 设置结点	2023-08-02
3	V1.0.2	添加 5.4.9 和 5.4.10 章节及 getRawImage 可选流程三 添加相机控制的字符串属性设置描述 3.6.1 修改 1.3 章节的控制流图，添加字符控制流程	2024-04-18