1 注意

请认真阅读该文档，里面包含了SDK提供的内容，使用方法，注意事项以及常见问题等，可以帮助开发人员快速的熟悉本SDK，如有疑问请联系技术支持人员。

【Please read this document carefully. It includes the content provided by the SDK, usage methods, precautions, and frequently asked questions. It can help developers quickly familiarize themselves with the SDK. If you have any questions, please contact technical support personnel.】

2 提供文件【Provided Files】

主要介绍本SDK提供的内容，以及各部分内容的用途，可以帮助开发人员快速的熟悉和上手，请认真阅读。

【It mainly introduces the content provided by this SDK and the purpose of each part of the content, which can help developers get familiar with and get started quickly. Please read it carefully.】

2.1 libirxxx_sdk_release

libxxx.aar

SDK对应的aar包【SDK corresponding aar package】

具体使用方式请参考`libir_sample`。【For specific usage, please refer to `libir_sample`.】

2.2 libirxxx_sample

SDK的参考demo，该demo是API接口的展示示例，仅供参考。

【The reference demo of the SDK, the demo is a demonstration example of the API interface, for reference only.】

注意【Note】：

目录下有`README.md`文件，其中详细介绍了出图流程，使用说明，注意事项以及常见问题解析，请认真阅读。

【There is a `README.md` file in the directory, which introduces the drawing process, instructions, precautions and common problem analysis in detail, please read it carefully.】

2.3 libirxxx_apk

SDK的参考demo对应的apk。

【SDK reference demo corresponding apk.】

您可以在您的设备上安装该apk以快速体验和测试，也可以作为对比方便排查开发中的一些问题。

【You can install the apk on your device for quick experience and testing, and it can also be used as a comparison to troubleshoot some problems in development.】

2.4 用户参考等效大气透过率查找表【LUT for user reference】

高增益和低增益下的测温距离修正表。不同的模组使用的距离修正表会有差异，这里给出的修正表和`libir_sample`目录中`assets/tau`下的修正表均为示例，您具体使用的修正表请联系对应人员确认。

【Temperature measurement distance correction table at high gain and low gain. The distance correction table used by different modules will be different. The correction table given here and the correction table under `assets/tau` in the `libir_sample` directory are examples. Please contact the corresponding personnel to confirm the specific correction table you use.】

2.5 接口开发说明文档【Interface development instructions】

Android端接口对应的开发说明文档。

【Development documentation corresponding to the interface on the Android side.】

主要讲解本SDK提供的工具类，API接口以及一些复杂功能的介绍，请认真阅读。

【It mainly explains the tool classes, API interfaces and some complex functions provided by this SDK, please read carefully.】

2.6 doc

库文件的接口说明文档。【Interface description documentation for library files.】

关于接口的注释，请求参数及其类型，返回值等请参考该文档。

【Please refer to this document for comments on the interface, request parameters and their types, return values, etc.】

注意：

在开发的过程中，如果对函数的使用方法，具体功能等有疑问，请在该文档中进行查询。

【During the development process, if you have any questions about how to use the function, specific functions, etc., please query in this document.】

2.7 用户开发标定【User calibration instructions】

用户进行锅盖标定、二次标定、盲元标定和环境变量修正的说明文件。

【Lid pattern noise correction，Secondary calibration， Dead pixel correction， Ambient variables correction parameters.】

2.7.1 测温与锅盖标定Secondary calibration& Lid pattern noise correction

测温和锅盖标定的具体使用说明，其中测温标定包含了单点标定，两点标定以及多点标定功能，在libxxx_sample中有对应的demo示例。

【Specific instructions for temperature measurement and pot lid calibration. The temperature measurement calibration includes single-point calibration, two-point calibration and multi-point calibration functions. There are corresponding demo examples in libxxx_sample.】

2.7.2 环境变量修正Ambient variable correction

环境变量修正的具体使用说明，其中包含了测温修正功能，在libxxx_sample中有对应的demo示例。

【The specific usage instructions for environment variable correction, which includes the temperature measurement correction function, has a corresponding demo example in libxxx_sample.】

2.7.3 新增盲元矫正Dead pixel correction

新增盲元以及去除盲元的具体使用说明，在libxxx_sample中有对应的demo示例。

【The specific instructions for adding blind elements and removing blind elements are provided in libxxx_sample.】

2.8 Tiny1c Android SPI Bring Up

SPI相关的调试文档。

【SPI related debugging documents.】

该文档只会在SPI协议的SDK中提供。

【This document will only be provided in the SDK of the SPI protocol.】

3 SDK简介 【Introduction to SDK】

Falcon Android SDK 实现了Android系统下Falcon系列红外摄像头应用接口，为客户提供参考demo和底层封装库方面客户二次开发，后面章节统一称为 SDK 。

【The Falcon Android SDK implements the Falcon series infrared camera application interface under the Android system, and provides customers with reference demos and the secondary development of the underlying package library. The following chapters are collectively referred to as SDK.】

4 SDK引入【SDK introduction】

4.1 步骤一【Step 1】

把aar库放在工程目录下【Put the aar library in the project directory】

4.2 步骤二【Step 2】

在build.gradle下添加库目录【Add the library directory under build.gradle】

```
dependencies {
    // use jar or aar
    implementation fileTree(dir: 'libs', include: ['*.jar','*.aar'])
    ...
}
```

5 APIs

SDK JAVA层提供Java类库`IRCMDType`，`UVCType`，`LibIRTemp`、`LibIRParse`、`LibIRProcess`、`IRCMD`、`UVCCamera`、`DualUVCCamera`，`IRUtils`，`CommonUtils`，`CommonParams`，`USBMonitor`。

【The SDK JAVA layer provides Java class libraries `IRCMDType`, `UVCType`, `LibIRTemp`, `LibIRParse`, `LibIRProcess`, `IRCMD`, `UVCCamera`, `DualUVCCamera`, `IRUtils`, `CommonUtils`, `CommonParams` , `USBMonitor`.】

每个库的作用如下：【The role of each library is as follows:】

备注：这里列出的API接口仅做了简单说明，方便用户对SDK的结构有大概了解，且不同的SDK使用到的具体接口会有差异。接口的请求参数或返回值，以及详细注释，请参考doc中的文档。

【Remarks: The API interfaces listed here are only briefly explained, so that users can have a general understanding of the structure of the SDK, and the specific interfaces used by different SDKs will be different. For the request parameters or return values of the interface, as well as detailed comments, please refer to the documentation in the doc.】

5.1 IRCMDType

机芯命令类型，需要区分USB和SPI类型。

【Movement command type, USB and SPI types need to be distinguished.】

在IRCMD初始化的时候会用到，请确认好您要使用的SDK类型以及对应的分辨率，不同的SDK类型接口和参数等可能会不同，不可混用。

【It will be used when IRCMD is initialized. Please confirm the type of SDK you want to use and the corresponding resolution. Different SDK types may have different interfaces and parameters, and cannot be mixed.】

5.2 IRCMD相关

与机芯固件进行通讯接口，实现信息读取和写入，实现机芯配置功能和固件升级等。

【It communicates with the movement firmware to realize information reading and writing, realize movement configuration functions and firmware upgrades, etc.】

SDK采用安卓软件库AAR格式发布。

【The SDK is released in the AAR format of the Android software library.】

5.2.1 初始化

```
..

// IRCMD init
ConcreteIRCMDBuilder concreteIRCMDBuilder = new ConcreteIRCMDBuilder();
            ircmd = concreteIRCMDBuilder
                    .setIrcmdType(IRCMDType.xxx)
                    .setIdCamera(uvcCamera.getNativePtr())
                    .build();
```

此初始化方式为标准SDK初始化方式，如有定制双向标定需求，请参考下面SDK和固件双向绑定文档。

5.2.2 接口

具体接口及返回值请参考`doc`中文档。【For specific interfaces and return values, please refer to the documentation in `doc`.】

5.3 UVCType

UVC类型，需要区分不同的协议类型。

【UVC type needs to distinguish different protocol types.】

在UVCCamera初始化的时候会用到，请确认好您所使用的SDK的协议类型，不同的协议不可混用。

【It will be used when UVCCamera is initialized. Please confirm the protocol type of the SDK you are using. Different protocols cannot be mixed.】

5.4 UVCCamera相关

用于UVC相机设备的寻找、设备插拔监听，连接打开和出图回调，以及发送对应的控制命令

【It is used to search for UVC camera equipment, monitor the plug-in and unplug of the equipment, open the connection and call back the picture, and send the corresponding control commands】

5.4.1 初始化

```
..

// UVCCamera init
ConcreateUVCBuilder concreateUVCBuilder = new ConcreateUVCBuilder();
uvcCamera = concreateUVCBuilder
            .setUVCType(UVCType.xxx)
            .bind();
```

设置不同的cameraWidth和cameraHeight可以控制不同的出图分辨率

```
uvcCamera.setUSBPreviewSize(cameraWidth, cameraHeight);
```

- cameraWidth:256,cameraHeight:384,红外+温度
- cameraWidth:256,cameraHeight:192,默认为红外

  调用startY16ModePreview接口，传入不同的参数可以获取不同的Y16格式数据流

5.4.2 接口

具体接口及返回值请参考`doc`中文档。【For specific interfaces and return values, please refer to the documentation in `doc`.】

5.5 DualUVCCamera相关

双光SDK会用到，单光SDK无需关系该函数。

【The dual-light SDK will use it, and the single-light SDK does not need to be related to this function.】

5.5.1 初始化

```
..

// DualUVCCamera 初始化
ConcreateDualBuilder concreateDualBuilder = new ConcreateDualBuilder();
dualUVCCamera = concreateDualBuilder
                .setDualType(DualType.xxx)
                .setIRSize(xxx, xxx)
                .setVLSize(xxx, xxx)
                .setDualSize(xxx, xxx)
                .setDataFlowMode(dataFlowMode)
                .setPreviewCameraStyle(CommonParams.PreviewCameraStyle.xxx)
                .setDeviceStyle(CommonParams.DeviceStyle.xxx)
                .build();
```

5.5.2 接口

具体接口及返回值请参考`doc`中文档。【For specific interfaces and return values, please refer to the documentation in `doc`.】

5.6 LibIRTemp相关

对温度数据的解析和处理，以及温度数据的二次修正等计算

【Analysis and processing of temperature data, and calculation of secondary correction of temperature data】

5.6.1 接口

具体接口及返回值请参考`doc`中文档。【For specific interfaces and return values, please refer to the documentation in `doc`.】

5.7 LibIRParse相关

对原始数据进行切分（为温度和图像），以及数据的格式转换，如RGB和YUV之间的转换

【Segment the original data (temperature and image), and data format conversion, such as conversion between RGB and YUV】

5.7.1 接口

具体接口及返回值请参考`doc`中文档。【For specific interfaces and return values, please refer to the documentation in `doc`.】

5.8 LibIRProcess相关

一些额外的处理算法，如伪彩色、旋转镜像翻转等

【Some additional processing algorithms, such as false color, rotating mirror flip, etc.】

5.8.1 接口

具体接口及返回值请参考`doc`中文档。【For specific interfaces and return values, please refer to the documentation in `doc`.】

5.9 IRUtils

红外相关的接口，用于设备离线的时候进行一些处理操作，适用于如图片二次编辑等场景。

【Infrared-related interfaces are used to perform some processing operations when the device is offline, and are suitable for scenarios such as secondary editing of pictures.】

5.10 CommonUtils

提供的一些公共函数，方便开发者快速使用。

【Some public functions provided are convenient for developers to use quickly.】

5.11 CommonParams

函数的公共参数类，提供参数的类型限制，避免使用错误。

【The public parameter class of the function, which provides the type restriction of the parameter to avoid usage errors.】

5.12 USBMonitor

用来监听USB设备的插拔，SPI协议无需该类。

【It is used to monitor the plugging and unplugging of USB devices, and the SPI protocol does not need this class.】

5.12.1 接口

具体接口及返回值请参考`doc`中文档。【For specific interfaces and return values, please refer to the documentation in `doc`.】

6 APP开发参考指南【APP Development Reference Guide】

6.1 APP出图和测温流程简介【Introduction to APP drawing and temperature measurement process】

- USBcamera处理类IRUVC实现摄像头插入状态监听、USB打开权限控制、摄像头参数初始化、摄像头连接预览命令和数据采集回调。在onConnect回调里面打开摄像头，在回调函数onFrame里面实现图像数据和温度数据获取拷贝分割和旋转等操作。

  【The USBcamera processing class IRUVC realizes the monitoring of camera insertion status, USB opening authority control, camera parameter initialization, camera connection preview command, and data collection callback. Turn on the camera in the onConnect callback, and implement operations such as image data and temperature data acquisition, copying, segmentation, and rotation in the callback function onFrame.】

- 图像数据经过回调采集送到ImageThread进行图像处理和转换操作得到ARGB格式。最后CameraView使用bitmap把ARGB格式数据放大填充屏幕进行图像绘制。

  【The image data is collected by callback and sent to ImageThread for image processing and conversion operations to obtain the ARGB format. Finally, CameraView uses bitmap to enlarge the ARGB format data to fill the screen for image drawing.】

- TemperatureView实现点、线、框温度测量显示，onTouch方法里面实现了手指绘制点、线、框并记录坐标，测温进程调用测温库实现了温度测量，最终显示在相应坐标。

  【TemperatureView realizes the temperature measurement and display of points, lines, and boxes. The onTouch method realizes the finger drawing points, lines, and boxes and records the coordinates. The temperature measurement process calls the temperature measurement library to realize the temperature measurement, and finally displays the corresponding coordinates.】

- 机芯配置命令，红外机芯固件提供一下接口实现图像和测试参数的配置，可以提高图像和测温质量。类Libircmd提供一系列native函数调用，需要在打开摄像头的情况下传入摄像头句柄指针调用。

  【Movement configuration command, infrared movement firmware provides the following interface to realize the configuration of image and test parameters, which can improve the quality of image and temperature measurement. The class Libircmd provides a series of native function calls, which need to be called with the camera handle pointer when the camera is turned on.】

6.2 设备pid过滤【Device pid filter】

UVC芯片pid会有默认值，如果您的设备pid不是该默认值，需要更改实际的pid过滤参数。

【UVC chip pid will have a default value. If your device pid is not the default value, you need to change the actual pid filter parameters.】

6.3 测温功能【Temperature measurement function】

6.3.1 Libirtemp库测温【Lipirtemp library temperature measurement】

- RAW温度数据摄氏度计算：【RAW temperature data Celsius calculation:】

  x y为坐标数据，cameraWidth为摄像头像素宽度，如果数据经过旋转，需要使用旋转后的宽度。

  【x y is the coordinate data, cameraWidth is the pixel width of the camera, if the data is rotated, the rotated width needs to be used.】

```
(temperature[(x*cameraWidth+y)*2]+((int)(temperature[(x*cameraWidth+y)*2+1])<<8))/64 -273.15;
```

- 测温库提供点线框测温接口【Temperature measurement library provides point wire frame temperature measurement interface】

调用Libirtemp的setTempData方法，传入温度数据，然后就可以调用对应的方法获取温度信息

```
/**
     * copy Temperature data from buffer<br/>
     *
     * @param src Temperature buffer
     */
    public void setTempData(byte[] src)
```

- 返回坐标点温度值摄氏度

【Returns the temperature value of the coordinate point in Celsius】

```
Libirtemp.getTemperatureOfPoint(Point point)
```

- 返回线的最低温、最高温、平均温、最低温坐标、最高温坐标

【Return line's lowest temperature, highest temperature, average temperature, lowest temperature coordinate, highest temperature coordinate】

```
Libirtemp.getTemperatureOfLine(Line line)
```

- 返回矩形的最低温、最高温、平均温、最低温坐标、最高温坐标

【Return the lowest temperature, highest temperature, average temperature, lowest temperature coordinates, and highest temperature coordinates of the rectangle】

```
Libirtemp.getTemperatureOfRect(Rect rect)
```

6.3.2 IRCMD库测温【IRCMD library temperature measurement】

- 返回坐标点温度值摄氏度

【Returns the temperature value of the coordinate point in Celsius】

```
public int getPointTemperatureInfo(int pixelPointX, int pixelPointY, int[]
temperatureValue)
```

- 返回线的最低温、最高温、平均温、最低温坐标、最高温坐标

【Return line's lowest temperature, highest temperature, average temperature, lowest temperature coordinate, highest temperature coordinate】

```
public int getLineTemperatureInfo(int startPointX, int startPointY, int
endPointX, int endPointY, int[] temperatureValue)
```

- 返回矩形的最低温、最高温、平均温、最低温坐标、最高温坐标

【Return the lowest temperature, highest temperature, average temperature, lowest temperature coordinates, and highest temperature coordinates of the rectangle】

```
public int getRectTemperatureInfo(int startPointX, int startPointY, int
endPointX, int endPointY,
                                   int[] temperatureValue)
```

- 获取整帧最小温度【Get the min temperature of the frame】

```
public int getCurrentFrameMinTemperature(int[] temperatureValue)
```

- 获取整帧最大温度【Get the maximum temperature of the frame】

```
public int getCurrentFrameMaxTemperature(int[] temperatureValue)
```

- 获取整帧的最大最小温度信息【Get the maximum and minimum temperature information of the frame】

```
public int getCurrentFrameMaxAndMinTemperature(int[] temperatureValue)
```

6.4 测温二次修正【Second correction of temperature measurement】

具体内容见文档 `用户开发标定 User calibration instructions\环境变量修正Ambient variable correction\环境变量修正Ambient variable correction.pdf`

【For details, see the document `User calibration instructions\Ambient variable correction\Ambient variable correction.pdf`】

6.5 高低增益自动切换【High and low gain automatic switching】

```
...
private LibIRProcess.AutoGainSwitchInfo_t auto_gain_switch_info = new LibIRProcess.AutoGainSwitchInfo_t();
private LibIRProcess.GainSwitchParam_t gain_switch_param = new LibIRProcess.GainSwitchParam_t();
...

        // 自动增益切换参数auto gain switch parameter
        gain_switch_param.above_pixel_prop = 0.1f;    //用于high -> low gain,设备像素总面积的百分比
        gain_switch_param.above_temp_data = (int) ((130 + 273.15) * 16 * 4); //用于high -> low gain,高增益向低增益切换的触发温度
        gain_switch_param.below_pixel_prop = 0.95f;   //用于low -> high gain,设备像素总面积的百分比
        gain_switch_param.below_temp_data = (int) ((110 + 273.15) * 16 * 4);//用于low -> high gain,低增益向高增益切换的触发温度
        auto_gain_switch_info.switch_frame_cnt = 5 * 15; //连续满足触发条件帧数超过该阈值会触发自动增益切换(假设出图速度为15帧每秒，则5 * 15大概为5秒)
        auto_gain_switch_info.waiting_frame_cnt = 7 * 15;//触发自动增益切换之后，会间隔该阈值的帧数不进行增益切换监测(假设出图速度为15帧每秒，则7 * 15大概为7秒)

...

/**
     * 高低增益自动切换【High and low gain automatic switching】<br/>
     * 每一帧都进行处理<br/>
     *
     * @param temp_frame             原始的温度数据<br/>
     * @param image_res              原始温度数据的宽高<br/>
     * @param auto_gain_switch_info  触发自动增益切换帧数阈值,当监测到在gain_switch_param条件下高增益或低增益的帧数超过auto_gain_switch_info时则触发自动增益切换<br/>
     * @param gain_switch_param      自动增益切换参数<br/>
     * @param autoGainSwitchCallback 自动切换回调
     * @return see {@link IrcmdResult}
     */
    public int autoGainSwitch(byte[] temp_frame, LibIRProcess.ImageRes_t image_res,
                              LibIRProcess.AutoGainSwitchInfo_t auto_gain_switch_info,
                              LibIRProcess.GainSwitchParam_t gain_switch_param,
                              AutoGainSwitchCallback autoGainSwitchCallback)
```

调用方式在IRUVC onframe回调里面每一帧调用即可。【The calling method can be called every frame in the IRUVC onframe callback.】

6.6 过曝保护【Overexposure protection】

```
...
// 防灼烧参数over_portect parameter
        int low_gain_over_temp_data = (int) ((550 + 273.15) * 16 * 4); //低增益下触发防灼烧的温度
        int high_gain_over_temp_data = (int) ((150 + 273.15) * 16 * 4); //高增益下触发防灼烧的温度
        float pixel_above_prop = 0.02f;//设备像素总面积的百分比
        int switch_frame_cnt = 7 * 15;//连续满足触发条件超过该阈值会触发防灼烧(假设出图速度为15帧每秒，则7 * 15大概为7秒)
        int close_frame_cnt = 10 * 15;//触发防灼烧之后，经过该阈值的帧数之后会解除防灼烧(假设出图速度为15帧每秒，则10 * 15大概为10秒)
...

/**
     * 过曝保护【Overexposure protection】<br/>
     * 每一帧都进行处理<br/>
     *
     * @param isUseIRISP                是否使用ISP算法，一般传false即可
     * @param gainStatus                当前的增益状态{@link CommonParams.GainStatus#HIGH_GAIN}{@link CommonParams.GainStatus#LOW_GAIN}<br/>
     * @param temp_frame                原始的温度数据<br/>
     * @param image_res                 原始温度数据的宽高<br/>
     * @param low_gain_over_temp_data   低增益下触发防灼烧温度值<br/>
     * @param high_gain_over_temp_data  高增益下触发防灼烧温度值<br/>
     * @param pixel_above_prop          pixels above the maxtemp,一帧中像素所占的比例大于该阈值会触发防灼烧<br/>
     * @param switch_frame_cnt          触发防灼烧的连续帧数<br/>
     * @param close_frame_cnt           触发防灼烧之后帧数达到该阈值之后会打开快门<br/>
     * @param avoidOverexposureCallback 防灼烧回调，true:触发防灼烧；false:防灼烧解除
     * @return see {@link IrcmdResult}
     */
    public int avoidOverexposure(boolean isUseIRISP, CommonParams.GainStatus gainStatus, byte[] temp_frame,                      LibIRProcess.ImageRes_t image_res, int low_gain_over_temp_data, int high_gain_over_temp_data, float pixel_above_prop,
 int switch_frame_cnt, int close_frame_cnt, AvoidOverexposureCallback avoidOverexposureCallback)
```

调用方式在IRUVC onframe回调里面每一帧调用即可。【The calling method can be called every frame in the IRUVC onframe callback.】

6.7 不同模式数据流【Different data flows】

切换不同的出图模式会输出不同分辨率和模式的数据流。

数据流【dataFlow】	数据【data】	分辨率【resolution】
IMAGE_AND_TEMP_OUTPUT	图像+温度【Image + Temp】	256*384
IMAGE_OUTPUT	图像【Image】	256*192

数据流【dataFlow】	数据【data】	分辨率【resolution】
TEMP_OUTPUT	温度(通过startY16ModePreview切换)【Temp(Toggle via startY16ModePreview)】	256*192

在`UVCCamera`初始化的时候，传入不同的分辨率会切换到不同的数据流模式：

```
uvcCamera = new UVCCamera(cameraWidth, cameraHeight);
/**
         * cameraWidth:256,cameraHeight:384,图像+温度
         * cameraWidth:256,cameraHeight:192,图像
         * cameraWidth:256,cameraHeight:192,(调用startY16ModePreview，传入Y16_MODE_TEMPERATURE)温度
         */
```

6.8 自定义伪彩

6.8.1 简介

为方便客户使用自定义的伪彩表，提供对应的伪彩表生成，格式转换，数据存储和读取等接口和示例。

6.8.2 生成自定义伪彩表

```
                // 生成自定义伪彩表
                int[][] color1 = new int[][]{{0}, {0, 0, 40}};
                int[][] color2 = new int[][]{{21}, {138, 20, 150}};
                int[][] color3 = new int[][]{{42}, {248, 135, 0}};
                int[][] color4 = new int[][]{{209}, {208, 48, 75}};
                int[][] color5 = new int[][]{{230}, {249, 143, 0}};
                int[][] color6 = new int[][]{{255}, {255, 255, 196}};
                byte[] pseudoDataByte = CommonUtils.generatePseudocolorData(color1, color2, color3, color4, color5, color6);
```

`color1, color2, color3, color4, color5, color6`为采样点，最少为两个，数量越多，效果越好。

6.8.3 伪彩表格式转换

对生成的伪彩表进行格式转换，以适用不同的场景。

- 转换为RGB伪彩

```
CommonUtils.convertRGBPseudocolorData
```

- 转换为YUV伪彩

```
CommonUtils.convertYUVPseudocolorData
```

6.8.4 使用自定义伪彩

默认使用系统伪彩，传入伪彩类型

```
LibIRProcess.convertYuyvMapToARGBPseudocolor
```

使用自定义伪彩，传入伪彩表数据

```
LibIRProcess.convertYuyvMapToARGBCustomPseudocolor
```

6.8.5 USB 双光自定义伪彩步骤

USB 双光使用自定义伪彩有所区别

6.8.5.1 step 1：

6.8.5.1.1 生成自定义伪彩表

```
                // 生成自定义伪彩表
                int[][] color1 = new int[][]{{0}, {0, 0, 40}};
                int[][] color2 = new int[][]{{21}, {138, 20, 150}};
                int[][] color3 = new int[][]{{42}, {248, 135, 0}};
                int[][] color4 = new int[][]{{209}, {208, 48, 75}};
                int[][] color5 = new int[][]{{230}, {249, 143, 0}};
                int[][] color6 = new int[][]{{255}, {255, 255, 196}};
                byte[] pseudoDataByte =
CommonUtils.generatePseudocolorData(color1, color2, color3, color4, color5,
color6);
```

`color1, color2, color3, color4, color5, color6` 为采样点，最少为两个，数量越多，效果越好。

6.8.5.2 step 2：

6.8.5.2.1 伪彩渲染

调用loadCustomPseudocolor和setCustomPseudocolor方法，伪彩名可自定义

```
dualView.getDualUVCCamera().loadCustomPseudocolor("custom", data);
dualView.getDualUVCCamera().setCustomPseudocolor("custom");
```