目录

一. 简介

1.1 产品概述

本 SDK 提供了和人脸相关的一些功能操作，包含人脸检测，人脸年龄检测，人脸性别检测，人脸三维角度信息检测，人脸识别。

1.2 环境要求

1.2.1 运行环境

Windows 7、CentOS7, Ubuntu14.04 以上的操作系统
JDK 8 或以上版本（注意区分 32 位和 64 位与算法库保持一致）
Windows 平台需安装 vs2013_runtime（注意区分 32 位和 64 位与算法库保持一致）
LINUX 平台需安装 GLIBC 2.17 库依赖及以上

1.2.2 支持的颜色空间格式

NV21, NV12, I420,YUYV,BGR24

常量名	常量值	常量说明
CP_PAF_NV21	2050	8-bit Y 通道，8-bit 2x2 采样 V 与 U 分量交织通道
ASVL_PAF_NV12	2049	8-bit Y 通道，8-bit 2x2 采样 U 与 V 分量交织通道
CP_PAF_BGR24	513	RGB 分量交织，按 B, G, R, B 字节序排布
ASVL_PAF_I420	1537	8-bit Y 通道， 8-bit 2x2 采样 U 通道， 8-bit 2x2 采样 V 通道
ASVL_PAF_YUYV	1289	YUV 分量交织， V 与 U 分量 2x1 采样，按 Y0, U0, Y1, V0 字节序排布

1.3 产品功能简介

1.3.1 人脸检测

对传入图像数据进行人脸检测，返回人脸位置信息和人脸在图像中的朝向信息，可用于后续的人脸分析、人脸比对操作，支持图像模式和视频流模式。

支持单人脸、多人脸检测，最多支持检测人脸数为 50。

1.3.2 人脸跟踪

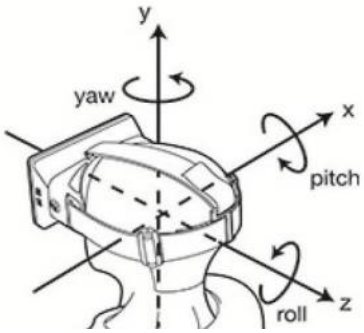
捕捉视频流中的人脸信息，并对人脸进行跟踪。

1.3.3 人脸属性检测

对检测到的人脸进行属性分析，支持性别、年龄的属性分析，支持图像模式和视频流模式。

1.3.4 人脸三维角度检测

检测输入图像数据指定区域人脸的三维角度信息，包含人脸三个空间角度：俯仰角（pitch）， 横滚角（roll）， 偏航角（yaw）， 支持图像模式和视频流模式。



1.3.5 人脸比对

将两个人脸进行比对，来判断是否为同一个人，返回比对相似度值。

1.3.6 活体检测

离线活体检测，基于 RGB 单目摄像头实现静默式识别。针对视频流/图片，通过采集人像的破绽来判断目标对象是否为活体，可有效防止照片、屏幕二次翻拍等作弊攻击。

1.4 SDK 授权说明

SDK 授权按设备进行授权，每台硬件设备需要一个独立的授权，此授权的校验基于设备的唯一标识，被授权的设备，初次授权时需要联网进行授权，授权成功后在有效期内可以离线运行 SDK。

激活一台设备后，遇以下情况，需要重新联网激活：

- 删除基于 SDK 开发的应用或删除应用数据
- 刷新系统
- 激活一台设备后，硬件设备变更

二. 接入指南

2.1 SDK 获取

2.1.1 注册为开发者

访问 ArcSoft AI 开放平台门户：https://ai.arcsoft.com.cn，注册开发者账号并登录。

2.1.2 SDK 下载

创建对应的应用，并选择需要下载的 SDK、对应平台即版本，确认后即可下载 SDK 和查看激活码。

点击【下载 SDK】即可下载 SDK 开发包；
点击【查看激活码】即可查看所需要 APPID、SDKKEY；

2.1.3 SDK 包结构

For Windows：32 位

```
|---doc
|    |---ARCSOFT_ARC_FACE_JAVA_DEVELOPER'S_GUIDE.pdf        开发说明文档
|    |---apidocs                                            API说明
|---libs
|    |---Win32
|    |    |---libarcsoft_face.dll                           引擎库
|    |    |---libarcsoft_face_engine.dll                    引擎库
|    |    |---libarcsoft_face_engine_jni.dll                引擎库
|    |--- arcsoft-sdk-face-2.1.0.0jar                       java依赖库
|---samplecode
|    |---FaceEngineTest.java                                示例工程
|---releasenotes.txt                                        更新说明
```

For Windows：64 位

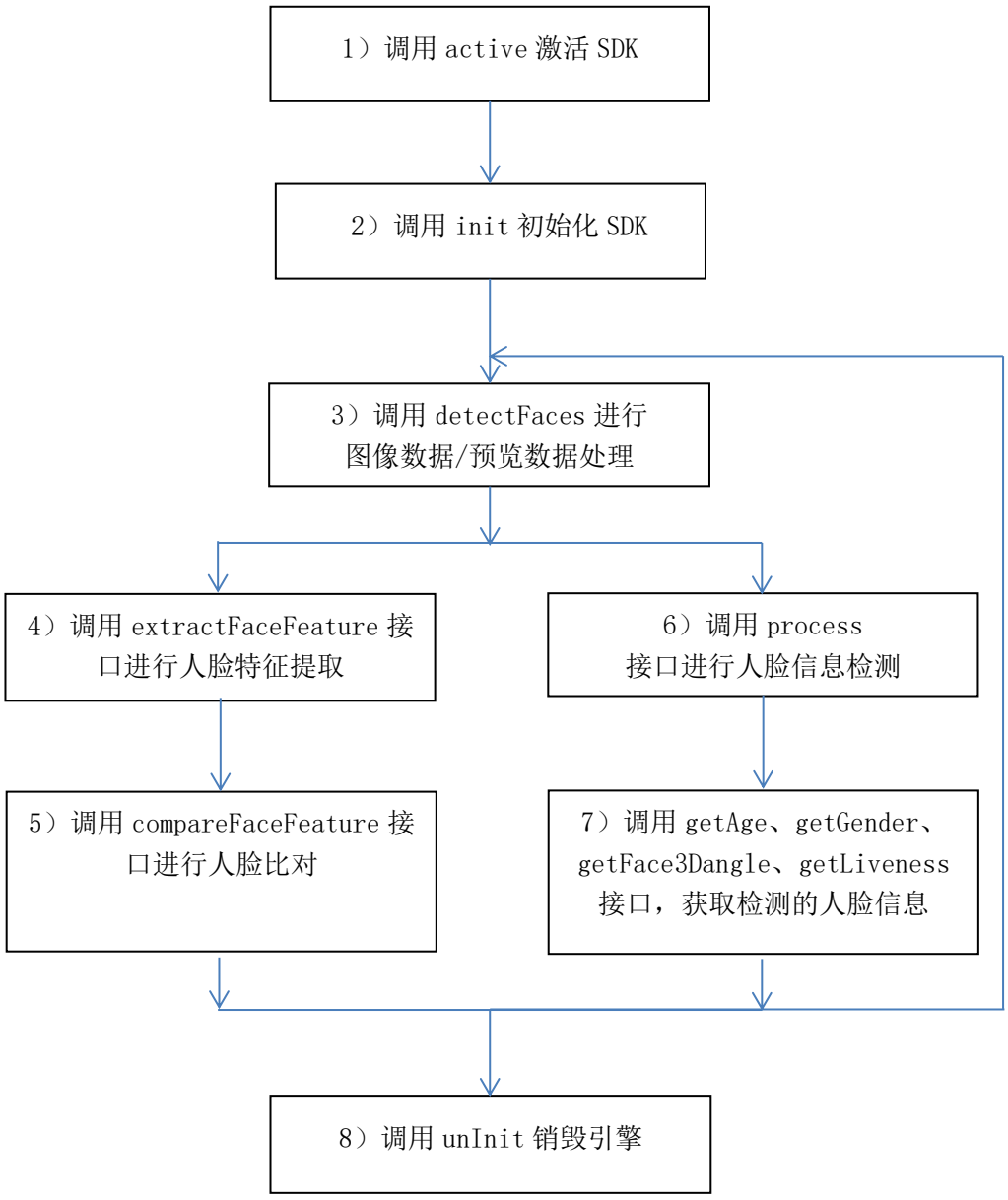
```
|---doc
|    |---ARCSOFT_ARC_FACE_JAVA_DEVELOPER'S_GUIDE.pdf        开发说明文档
|    |---apidocs                                            API说明
|---libs
|    |---Win64
|    |    |---libarcsoft_face.dll                           引擎库
|    |    |---libarcsoft_face_engine.dll                    引擎库
```

```
|   |   |---libarcsoft_face_engine_jni.dll                引擎库
|   |--- arcsoft-sdk-face-2.1.0.0jar                      java依赖库
|---samplecode
|   |---FaceEngineTest.java                               示例工程
|---releasenotes.txt                                      更新说明
```

For Linux：64 位

```
|---doc
|   |---ARCSOFT_ARC_FACE_JAVA_DEVELOPER'S_GUIDE.pdf       开发说明文档
|   |---apidocs                                           API说明
|---libs
|   |--- Linux64
|   |   |---libarcsoft_face.so                            引擎库
|   |   |---libarcsoft_face_engine.so                     引擎库
|   |   |---libarcsoft_face_engine_jni.so                 引擎库
|   |--- arcsoft-sdk-face-2.1.0.0jar                      java依赖库
|---samplecode
|   |---FaceEngineTest.java                               示例工程
|---releasenotes.txt                                      更新说明
```

三. SDK 调用流程



Step1：调用 FaceEngine 的 active 函数激活设备，一个设备安装后仅需激活一次，卸载重新安装后需要重新激活。

Step2：调用 FaceEngine 的 init 函数初始化 SDK，初始化成功后才能进一步使用 SDK 的功能。

Step3：调用 FaceEngine 的 detectFaces 函数进行图像数据或预览数据的人脸检测，若检测成功，则可得到一个人脸列表。（初始化时需要 supportFaceDetect）

Step4：调用 FaceEngine 的 extractFaceFeature 可对图像中指定的人脸进行特征提取。（初始化时需要 supportFaceRecognition）

Step5：调用 FaceEngine 的 compareFaceFeature 可对传入的两个人脸特征进行比对，获取相似度。（初始化时需要 supportFaceRecognition）

Step6：调用 FaceEngine 的 process，配置 FunctionConfiguration 可对 Age、Gender、Face3Dangle、Liveness 进行检测，传入的 FunctionConfiguration 的任一属性都需要在 init 时进行初始化。

Step7：调用 FaceEngine 的 getAge、getGender、getFace3Dangle、getLiveness 接口可获取年龄、性别、三维角度、活体信息结果，且每个结果在获取前都需要在 process 中进行处理。

Step8:调用 FaceEngine 的 unInit 销毁引擎。在 init 成功后如不 unInit 会导致内存泄漏。

四. 常见问题

4.1 错误码概览

错误码名	十六进制	十进制	错误码说明
MOK	0	0	成功
MERR_UNKNOWN	1	1	错误原因不明
MERR_INVALID_PARAM	2	2	无效的参数
MERR_UNSUPPORTED	3	3	引擎不支持
MERR_NO_MEMORY	4	4	内存不足
MERR_BAD_STATE	5	5	状态错误
MERR_USER_CANCEL	6	6	用户取消相关操作
MERR_EXPIRED	7	7	操作时间过期
MERR_USER_PAUSE	8	8	用户暂停操作
MERR_BUFFER_OVERFLOW	9	9	缓冲上溢
MERR_BUFFER_UNDERFLOW	A	10	缓冲下溢
MERR_NO_DISKSPACE	B	11	存贮空间不足
MERR_COMPONENT_NOT_EXIST	C	12	组件不存在
MERR_GLOBAL_DATA_NOT_EXIST	D	13	全局数据不存在
MERR_FSDK_INVALID_APP_ID	7001	28673	无效的 App Id
MERR_FSDK_INVALID_SDK_ID	7002	28674	无效的 SDK key
MERR_FSDK_INVALID_ID_PAIR	7003	28675	AppId 和 SDKKey 不匹配
MERR_FSDK_MISMATCH_ID_AND_SDK	7004	28676	SDKKey 和使用的 SDK 不匹配

MERR_FSDK_SYSTEM_VERSION_UNSUPPORTED	7005	28677	系统版本不被当前 SDK 所支持
MERR_FSDK_LICENCE_EXPIRED	7006	28678	SDK 有效期过期，需要重新下载更新
MERR_FSDK_FR_INVALID_MEMORY_INFO	12001	73729	无效的输入内存
MERR_FSDK_FR_INVALID_IMAGE_INFO	12002	73730	无效的输入图像参数
MERR_FSDK_FR_INVALID_FACE_INFO	12003	73731	无效的脸部信息
MERR_FSDK_FR_NO_GPU_AVAILABLE	12004	73732	当前设备无 GPU 可用
MERR_FSDK_FR_MISMATCHED_FEATURE_LEVEL	12005	73733	待比较的两个人脸特征的版本不一致
MERR_FSDK_FACEFEATURE_UNKNOWN	14001	81921	人脸特征检测错误未知
MERR_FSDK_FACEFEATURE_MEMORY	14002	81922	人脸特征检测内存错误
MERR_FSDK_FACEFEATURE_INVALID_FORMAT	14003	81923	人脸特征检测格式错误
MERR_FSDK_FACEFEATURE_INVALID_PARAM	14004	81924	人脸特征检测参数错误
MERR_FSDK_FACEFEATURE_LOW_CONFIDENCE_LEVEL	14005	81925	人脸特征检测结果置信度低
MERR_ASF_EX_FEATURE_UNSUPPORTED_ON_INIT	15001	86017	Engine 不支持的检测属性
MERR_ASF_EX_FEATURE_UNINITED	15002	86018	需要检测是属性未初始化
MERR_ASF_EX_FEATURE_UNPROCESSED	15003	86019	待获取的属性未在 process 中处理过
MERR_ASF_EX_FEATURE_UNSUPPORTED_ON_PROCESS	15004	86020	PROCESS 不支持的检测属性，例如 FR，有自己独立的处理函数
MERR_ASF_EX_INVALID_IMAGE_INFO	15005	86021	无效的输入图像
MERR_ASF_EX_INVALID_FACE_INFO	15006	86022	无效的脸部信息
MERR_ASF_ACTIVATION_FAIL	16001	90113	SDK 激活失败,请打开读写权限
MERR_ASF_ALREADY_ACTIVATED	16002	90114	SDK 已激活
MERR_ASF_NOT_ACTIVATED	16003	90115	SDK 未激活
MERR_ASF_SCALE_NOT_SUPPORT	16004	90116	detectFaceScaleVal 不支持

MERR_ASF_VERION_MISMATCH	16005	90117	SDK 版本不匹配
MERR_ASF_DEVICE_MISMATCH	16006	90118	设备不匹配
MERR_ASF_UNIQUE_IDENTIFIER_MISMATCH	16007	90119	唯一标识不匹配
MERR_ASF_PARAM_NULL	16008	90120	参数为空
MERR_ASF_LIVENESS_EXPIRED	16009	90121	活体检测功能已过期
MERR_ASF_VERSION_NOT_SUPPORT	1600A	90122	版本不支持
MERR_ASF_SIGN_ERROR	1600B	90123	签名错误
MERR_ASF_DATABASE_ERROR	1600C	90124	数据库插入错误
MERR_ASF_UNIQUE_CHECKOUT_FAIL	1600D	90125	唯一标识符校验失败
MERR_ASF_COLOR_SPACE_NOT_SUPPORT	1600E	90126	颜色空间不支持
MERR_ASF_IMAGE_WIDTH_HEIGHT_NOT_SUPPORT	1600F	90127	图片宽度或高度不支持
MERR_ASF_READ_PHONE_STATE_DENIED	16010	90128	android.permission.READ_PHONE_STATE 权限被拒绝
MERR_ASF_ACTIVATION_DATA_DESTROYED	16011	90129	激活数据被破坏,请删除激活文件,重新进行激活
MERR_ASF_SERVER_UNKNOWN_ERROR	16012	90130	服务端未知错误
MERR_ASF_NETWORK_COULDNT_RESOLVE_HOST	17001	94209	无法解析主机地址
MERR_ASF_NETWORK_COULDNT_CONNECT_SERVER	17002	94210	无法连接服务器
MERR_ASF_NETWORK_CONNECT_TIMEOUT	17003	94211	网络连接超时
MERR_ASF_NETWORK_UNKNOWN_ERROR	17004	94212	网络未知错误

4.2 FAQ

Q：如何将人脸识别 1:1 进行开发改为 1:n？

A: 先将人脸特征数据用本地文件、数据库或者其他的方式存储下来，若检测出结果需

要显示图像可以保存对应的图像。之后循环对特征值进行对比，相似度最高者若超过您设置的阈值则输出相关信息。

Q: 初始化引擎时检测方向应该怎么选择？

A: SDK 初始化引擎中可选择仅对 0 度、90 度、180 度、270 度单角度进行人脸检测，也可选择全角度进行检测；根据应用场景，推荐使用单角度进行人脸检测，因为选择全角度的情况下，算法中会对每个角度检测一遍，导致性能相对于单角度较慢。

Q: 初始化引擎时（detectFaceScaleVal）参数多大比较合适？

A: 用于数值化表示的最小人脸尺寸，该尺寸代表人脸尺寸相对于图片长边的占比。video 模式有效值范围[2,16], Image 模式有效值范围[2,32]，多数情况下推荐值为 16，特殊情况下可根据具体场景下进行设置；

Q: 初始化引擎之后调用其他接口返回错误码 86018，该怎么解决？

A: 86016 即需要检测的属性未初始化，需要查看调用接口的宏有没有在初始化引擎时在 combineMask 参数中加入。

Q: 调用 detectFaces、extractFaceFeature 和 process 接口返回 90127 错误码，该怎么解决？

A: SDK 对图像尺寸做了限制，宽高大于 0，宽度为 4 的倍数，YUYV/I420/NV21/NV12 格式的图片高度为 2 的倍数，BGR24 格式的图片高度不限制；如果遇到 90127 请检查传入的图片尺寸是否符合要求，若不符合可对图片进行适当的裁剪。

Q: 人脸检测结果的人脸框 Rect 为何有时会溢出传入图像的边界？

A: Rect 溢出边界可能是人脸只有一部分在图像中，算法会对人脸的位置进行估计。

Q: 为何调用引擎有时会出现 crash？

A: 若在引擎调用过程中进行销毁引擎则可能会导致 crash。在使用过程中应避免在销毁引擎时还在使用引擎，尤其是做特征提取或活体检测等耗时操作时销毁引擎，如加锁解决。

Q: MERR_FSDK_FACEFEATURE_LOW_CONFIDENCE_LEVEL，人脸检测结果置信度低是什么情况导致的？

A: 图片模糊或者传入的人脸框不正确。

Q：哪些因素会影响人脸检测、人脸跟踪、人脸特征提取等 SDK 调用所用时间？

A：硬件性能、图片质量等。

4.3 其他帮助

SDK 交流论坛：http://ai.arcsoft.com.cn/bbs/

4.4 示例代码

```
import com.arcsoft.face.*;
import com.arcsoft.face.enums.ImageFormat;
import javax.imageio.ImageIO;
import java.awt.color.ColorSpace;
import java.awt.image.BufferedImage;
import java.awt.image.ColorConvertOp;
import java.awt.image.DataBufferByte;
import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class FaceEngineTest {

    public static void main(String[] args) {
        new FaceEngineTest().faceEngineTest();
    }


    public void faceEngineTest() {
        String appId = "";
        String sdkKey = "";
```

```
        FaceEngine faceEngine = new FaceEngine();
        //激活引擎
        faceEngine.active(appId, sdkKey);
        EngineConfiguration engineConfiguration =
EngineConfiguration.builder().functionConfiguration(
                FunctionConfiguration.builder()
                        .supportAge(true)
                        .supportFace3dAngle(true)
                        .supportFaceDetect(true)
                        .supportFaceRecognition(true)
                        .supportGender(true)
                        .supportLiveness(true)
                        .build()).build();
        //初始化引擎
        faceEngine.init(engineConfiguration);

        ImageInfo imageInfo = getRGBData(new File("d:\\1.jpg"));
        ImageInfo imageInfo2 = getRGBData(new File("d:\\1.jpg"));
        //人脸检测
        List<FaceInfo> faceInfoList = new ArrayList<FaceInfo>();
        faceEngine.detectFaces(imageInfo.getRgbData(),
imageInfo.getWidth(), imageInfo.getHeight(),
ImageFormat.CP_PAF_BGR24, faceInfoList);

        //提取人脸特征
        FaceFeature faceFeature = new FaceFeature();
        faceEngine.extractFaceFeature(imageInfo.getRgbData(),
imageInfo.getWidth(), imageInfo.getHeight(),
ImageFormat.CP_PAF_BGR24, faceInfoList.get(0), faceFeature);

        FaceFeature faceFeature2 = new FaceFeature();
        faceEngine.extractFaceFeature(imageInfo2.getRgbData(),
imageInfo2.getWidth(), imageInfo2.getHeight(),
ImageFormat.CP_PAF_BGR24, faceInfoList.get(0), faceFeature2);

        //人脸对比
        FaceFeature targetFaceFeature = new FaceFeature();

targetFaceFeature.setFeatureData(faceFeature.getFeatureData());

        FaceFeature sourceFaceFeature = new FaceFeature();

sourceFaceFeature.setFeatureData(faceFeature2.getFeatureData());
```

```
        FaceSimilar faceSimilar = new FaceSimilar();
        faceEngine.compareFaceFeature(targetFaceFeature,
sourceFaceFeature, faceSimilar);

        int processResult =
faceEngine.process(imageInfo.getRgbData(), imageInfo.getWidth(),
imageInfo.getHeight(), ImageFormat.CP_PAF_BGR24, faceInfoList,
FunctionConfiguration.builder().supportAge(true).supportFace3dAngle(t
rue).supportGender(true).supportLiveness(true).build());
        //性别提取
        List<GenderInfo> genderInfoList = new
ArrayList<GenderInfo>();
        int genderCode = faceEngine.getGender(genderInfoList);
        //年龄提取
        List<AgeInfo> ageInfoList = new ArrayList<AgeInfo>();
        int ageCode = faceEngine.getAge(ageInfoList);
        //3D 信息提取
        List<Face3DAngle> face3DAngleList = new
ArrayList<Face3DAngle>();
        int face3dCode = faceEngine.getFace3DAngle(face3DAngleList);
        //活体信息
        List<LivenessInfo> livenessInfoList = new
ArrayList<LivenessInfo>();
        int livenessCode = faceEngine.getLiveness(livenessInfoList);
        System.out.println();
    }

    public ImageInfo getRGBData(File file) {
        if (file == null)
            return null;
        ImageInfo imageInfo;
        try {
            //将图片文件加载到内存缓冲区
            BufferedImage image = ImageIO.read(file);
            imageInfo = bufferedImage2ImageInfo(image);
        } catch (IOException e) {
            e.printStackTrace();
            return null;
        }
        return imageInfo;
    }

    private ImageInfo bufferedImage2ImageInfo(BufferedImage image) {
        ImageInfo imageInfo = new ImageInfo();
```

```
        int width = image.getWidth();
        int height = image.getHeight();
        // 使图片居中
        width = width & (~3);
        height = height & (~3);
        imageInfo.setWidth(width);
        imageInfo.setHeight(height);
        //根据原图片信息新建一个图片缓冲区
        BufferedImage resultImage = new BufferedImage(width, height,
image.getType());
        //得到原图的 rgb 像素矩阵
        int[] rgb = image.getRGB(0, 0, width, height, null, 0,
width);
        //将像素矩阵 绘制到新的图片缓冲区中
        resultImage.setRGB(0, 0, width, height, rgb, 0, width);
        //进行数据格式化为可用数据
        BufferedImage dstImage = new BufferedImage(width, height,
BufferedImage.TYPE_3BYTE_BGR);
        if (resultImage.getType() != BufferedImage.TYPE_3BYTE_BGR) {
            ColorSpace cs =
ColorSpace.getInstance(ColorSpace.CS_LINEAR_RGB);
            ColorConvertOp colorConvertOp = new ColorConvertOp(cs,
dstImage.createGraphics().getRenderingHints());
            colorConvertOp.filter(resultImage, dstImage);
        } else {
            dstImage = resultImage;
        }
        //获取 rgb 数据
        imageInfo.setRgbData(((DataBufferByte)
(dstImage.getRaster().getDataBuffer())).getData());
        return imageInfo;
    }


    class ImageInfo {
        public byte[] rgbData;
        public int width;
        public int height;
        public byte[] getRgbData() {
            return rgbData;
        }
        public void setRgbData(byte[] rgbData) {
            this.rgbData = rgbData;
        }
```

```
        public int getWidth() {
            return width;
        }
        public void setWidth(int width) {
            this.width = width;
        }
        public int getHeight() {
            return height;
        }
        public void setHeight(int height) {
            this.height = height;
        }
    }
}
```