ArcSoft ArcFace SDK

开发说明文档

目录

1.简介

1.1 产品概述

ArcFace 离线 SDK，包含人脸检测、性别检测、年龄检测、人脸识别、活体检测等能力，初次使用时需联网激活，激活后即可本地无网络环境下工作，可根据业务需求结合人脸识别等 SDK 灵活地进行应用层开发。

1.2 环境要求

1.2.1 运行环境

Android armeabi-v7a

1.2.2 系统要求

支持 Android 4.4 (API Level 19)及以上系统。

1.2.3 权限申明

获取设备唯一标识，用于 SDK 激活授权
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
允许应用联网，用于 SDK 联网激活授权
<uses-permission android:name="android.permission.INTERNET" />

1.2.4 支持的颜色空间格式

NV21, BGR24

常量名	常量值	常量说明
CP_PAF_NV21	2050	8-bit Y 层，之后是 8-bit 的 2x2 采样的 U，V 交织层
CP_PAF_BGR24	513	第一个字节为 R，第二个字节为 G，第三个字节为 B

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

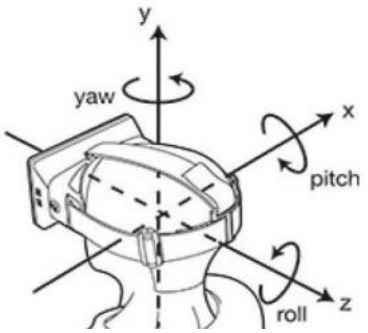
捕捉视频流中的人脸信息，并对人脸进行跟踪。

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
- 刷新安卓系统
- 激活一台设备后，硬件信息发生变更

2.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；

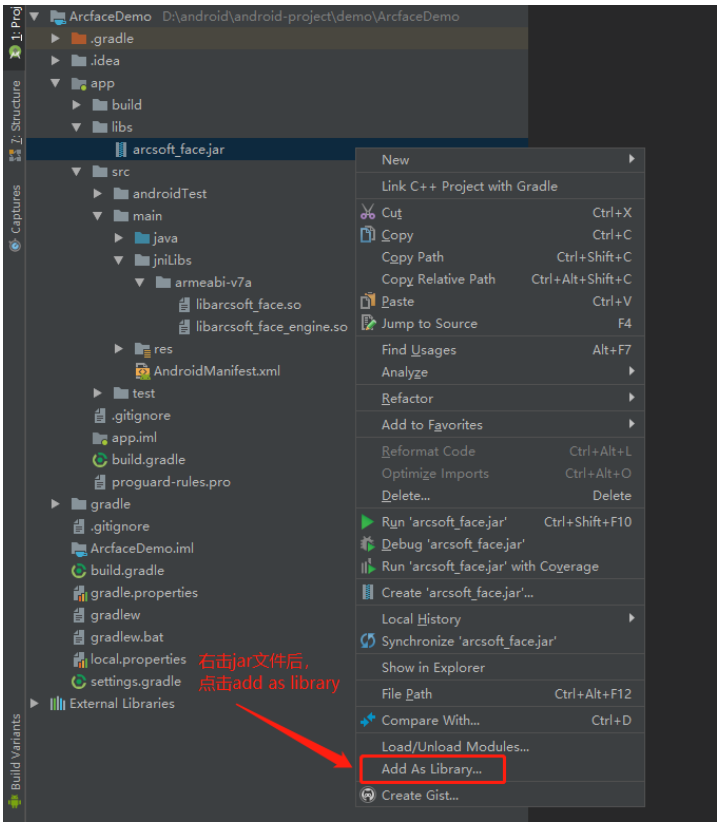
2.1.3 SDK 包结构

```
|---doc
|    |---ARCSOFT_ARCFACE_DEVELOPER'S_GUIDE.pdf        开发说明文档
|    |---ArcFaceAPI-Android.zip                       API说明
|---libs
|    |---armeabi-v7a
|    |    |---libarcsoft_face.so                       引擎库
|    |    |---libarcsoft_face_engine.so                引擎库
|    |---arcsoft_face.jar                              引擎库
|---samplecode
|    |---ArcfaceDemo                                   示例工程
|---releasenotes.txt                                   更新说明
```

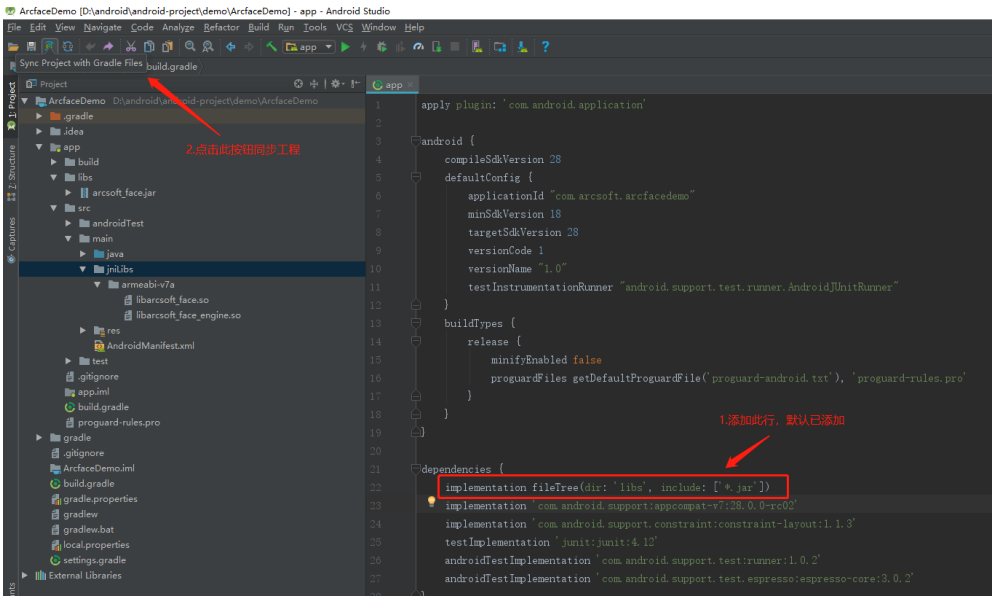
2.1.4 工程配置

1. 新建一个 Android Project，切换到 Project 视图；
2. 将 libarcsoft_face.so 和 libarcsoft_face_engine.so 添加到 src->main->jniLibs->armeabi-v7a 路径下；
3. 将 arcsoft_face.jar 放入，并依赖进工程；

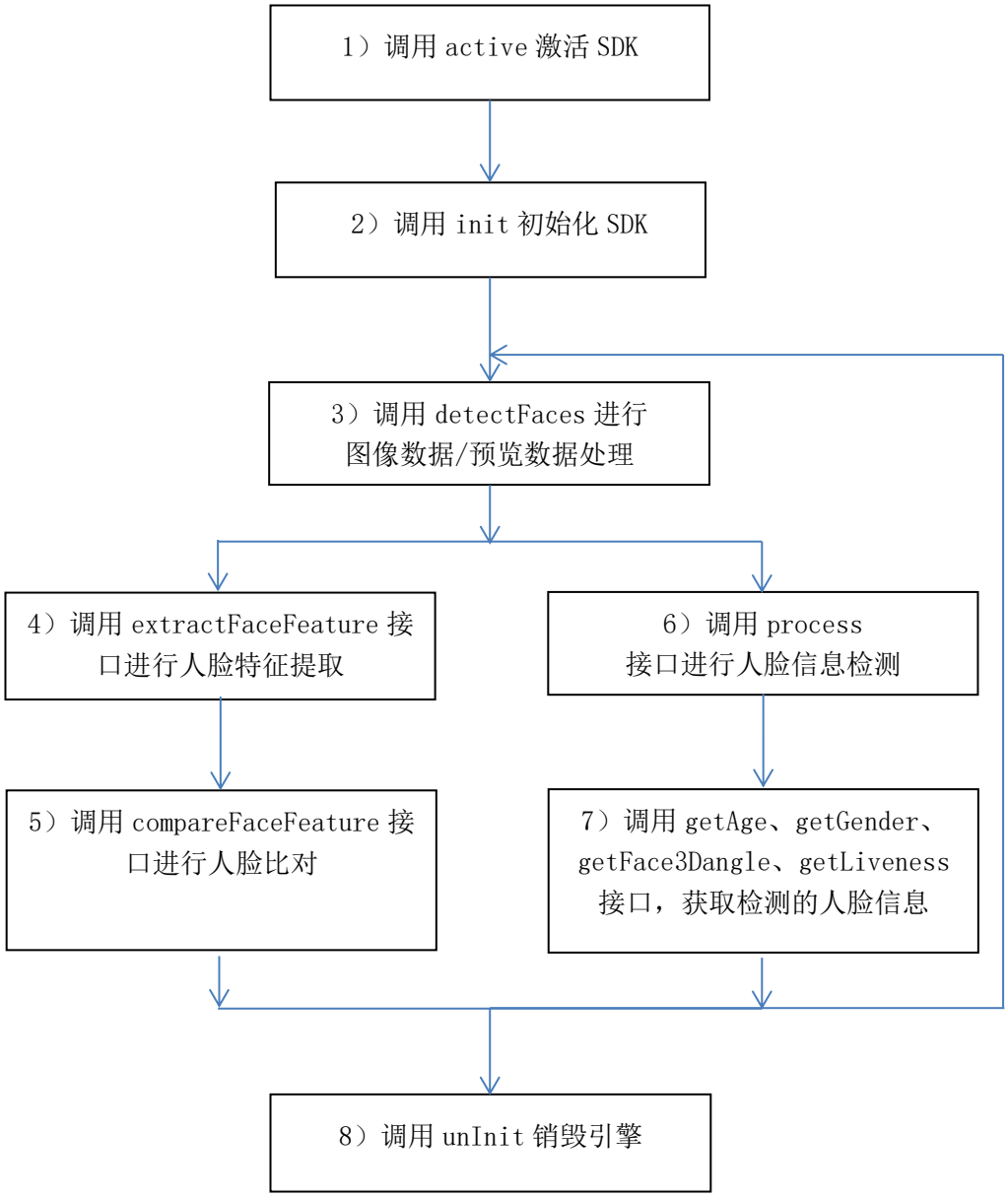
方法 1：


ArcfaceDemo D:\android\android-project\demo\ArcfaceDemo
.gradle
.idea
app
build
libs
arcsoft_face.jar
src
androidTest
main
java
jniLibs
armeabi-v7a
libarcsoft_face.so
libarcsoft_face_engine.so
res
AndroidManifest.xml
test
.gitignore
app.iml
build.gradle
proguard-rules.pro
gradle
.gitignore
ArcfaceDemo.iml
build.gradle
gradle.properties
gradlew
gradlew.bat
local.properties
settings.gradle
External Libraries
New
Link C++ Project with Gradle
Cut Ctrl+X
Copy Ctrl+C
Copy Path Ctrl+Shift+C
Copy Relative Path Ctrl+Alt+Shift+C
Paste Ctrl+V
Jump to Source F4
Find Usages Alt+F7
Analyze
Refactor
Add to Favorites
Reformat Code Ctrl+Alt+L
Optimize Imports Ctrl+Alt+O
Delete... Delete
Run 'arcsoft_face.jar' Ctrl+Shift+F10
Debug 'arcsoft_face.jar'
Run 'arcsoft_face.jar' with Coverage
Create 'arcsoft_face.jar'...
Local History
Synchronize 'arcsoft_face.jar'
Show in Explorer
File Path Ctrl+Alt+F12
Compare With... Ctrl+D
Load/Unload Modules...
Add As Library...
Create Gist...
右击jar文件后，
点击add as library

方法 2：


ArcfaceDemo [D:\android\android-project\demo\ArcfaceDemo] - app - Android Studio
Sync Project with Gradle Files
2.点击此按钮同步工程
1.添加此行，默认已添加
apply plugin: 'com.android.application'
android {
compileSdkVersion 28
defaultConfig {
applicationId "com.arcsoft.arcfacedemo"
minSdkVersion 18
targetSdkVersion 28
versionCode 1
versionName "1.0"
testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
}
buildTypes {
release {
minifyEnabled false
proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
}
}
}
dependencies {
implementation fileTree(dir: 'libs', include: ['*.jar'])
implementation 'com.android.support:appcompat-v7:28.0.0-rc02'
implementation 'com.android.support.constraint:constraint-layout:1.1.3'
testImplementation 'junit:junit:4.12'
androidTestImplementation 'com.android.support.test:runner:1.0.2'
androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'

2.1.5 调用流程



Step1：调用 FaceEngine 的 active 方法激活设备，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 方法初始化 SDK，初始化成功后才能进一步使用 SDK 的功能。

Step3：调用 FaceEngine 的 detectFaces 方法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 combineMask 需要 ASF_FACE_DETECT）

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

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

Step6：调用 FaceEngine 的 process 方法，传入不同的 combineMask 组合可对 Age、Gender、Face3Dangle、Liveness 进行检测，传入的 combineMask 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 方法可获取年龄、性别、三维角度、活体检测结果，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 方法销毁引擎。在 init 成功后如不 unInit 会导致内存泄漏。

2.2 核心类介绍

2.2.1 Class FaceEngine

类描述：

人脸引擎类，其中定义了人脸操作相关的函数，包含 SDK 的授权激活、引擎初始化以及人脸处理相关方法。

具体说明见 JavaDoc。

2.2.2 Class FaceInfo

类描述：

人脸信息类，用于存储人脸框以及人脸在图片中的朝向信息。

具体说明见 JavaDoc。

2.2.3 Class AgeInfo

类描述：

年龄信息类，用于存储年龄信息。

具体说明见 JavaDoc。

2.2.4 Class GenderInfo

类描述：

性别信息类，用于存储性别信息。

具体说明见 JavaDoc。

2.2.5 Class Face3DAngle

类描述：

人脸三维角度信息类，用于存储人脸三维角度信息。

具体说明见 JavaDoc。

2.2.6 Class LivenessInfo

类描述：

活体信息类，用于存储活体信息。

具体说明见 JavaDoc。

2.2.7 Class FaceFeature

类描述：

人脸特征信息类，用于存储人脸特征信息。

具体说明见 JavaDoc。

2.2.8 Class FaceSimilar

类描述：

人脸相似度信息类，用于人脸相似度信息。

具体说明见 JavaDoc。

2.2.9 Class VersionInfo

类描述：

版本信息类，用于存储版本信息。

具体说明见 JavaDoc。

2.2.10Class ErrorInfo

类描述：

错误信息类，其中定义了引擎调用的错误码。

具体说明见 JavaDoc。

2.3 错误代码

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（注意：调用初始化引擎接口时，请确认激活接口传入的参数，并重新激活）

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，写入激活文件失败
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_ACTIVEFILE_SDKTYPE_MISMATCH	16005	90117	激活文件与 SDK 类型不匹配
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,请清除应用数据或卸载应用后，重新进行激活
MERR_ASF_SERVER_UNKNOWN_ERROR	16012	90130	服务端未知错误
MERR_ASF_INTERNET_DENIED	16013	90131	INTERNET 权限被拒绝
MERR_ASF_ACTIVEFILE_SDK_MISMATCH	16014	90132	激活文件与 SDK 版本不匹配,请重新激活
MERR_ASF_DEVICEINFO_LESS	16015	90133	设备信息太少，不足以生成设备指纹
MERR_ASF_REQUEST_TIMEOUT	16016	90134	客户端时间与服务器时间（即北京时间）前后相差在 30 分钟之内
MERR_ASF_APPID_DATA_DECRYPT	16017	90135	服务端解密失败
MERR_ASF_APPID_APPKEY_SDK_MISMATCH	16018	90136	传入的 AppId 和 AppKey 与使用的 SDK 版本不一致
MERR_ASF_NO_REQUEST	16019	90137	短时间大量请求会被禁止请求,30 分钟之后会解封
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

2.4 通用方法

2.4.1 Bitmap 转换成 NV21 数据

```
/**
 * Bitmap 转化为 ARGB 数据，再转化为 NV21 数据
 *
 * @param src    传入的 Bitmap，格式为 Bitmap.Config.ARGB_8888
 * @param width  NV21 图像的宽度
 * @param height NV21 图像的高度
 * @return nv21 数据
 */
public static byte[] bitmapToNv21(Bitmap src, int width, int height) {
    if (src != null && src.getWidth() >= width && src.getHeight() >= height) {
        int[] argb = new int[width * height];
        src.getPixels(argb, 0, width, 0, 0, width, height);
        return argbToNv21(argb, width, height);
    } else {
        return null;
    }
}

/**
 * ARGB 数据转化为 NV21 数据
 *
 * @param argb   argb 数据
 * @param width  宽度
 * @param height 高度
 * @return nv21 数据
 */
private static byte[] argbToNv21(int[] argb, int width, int height) {
    int frameSize = width * height;
    int yIndex = 0;
    int uvIndex = frameSize;
    int index = 0;
```

```
    byte[] nv21 = new byte[width * height * 3 / 2];
    for (int j = 0; j < height; ++j) {
        for (int i = 0; i < width; ++i) {
            int R = (argb[index] & 0xFF0000) >> 16;
            int G = (argb[index] & 0x00FF00) >> 8;
            int B = argb[index] & 0x0000FF;
            int Y = (66 * R + 129 * G + 25 * B + 128 >> 8) + 16;
            int U = (-38 * R - 74 * G + 112 * B + 128 >> 8) + 128;
            int V = (112 * R - 94 * G - 18 * B + 128 >> 8) + 128;
            nv21[yIndex++] = (byte) (Y < 0 ? 0 : (Y > 255 ? 255 : Y));
            if (j % 2 == 0 && index % 2 == 0 && uvIndex < nv21.length -
2) {
                nv21[uvIndex++] = (byte) (V < 0 ? 0 : (V > 255 ? 255 :
V));
                nv21[uvIndex++] = (byte) (U < 0 ? 0 : (U > 255 ? 255 :
U));
            }

            ++index;
        }
    }
    return nv21;
}
```

2.4.2 Bitmap 转换成 BGR 数据

```
/**
 * bitmap 转化为 bgr 数据，格式为 Bitmap.Config.ARGB_8888
 *
 * @param image 传入的 bitmap
 * @return bgr 数据
 */
public static byte[] bitmapToBgr(Bitmap image) {
    if (image == null) {
        return null;
    }
    int bytes = image.getByteCount();

    ByteBuffer buffer = ByteBuffer.allocate(bytes);
    image.copyPixelsToBuffer(buffer);
    byte[] temp = buffer.array();
    byte[] pixels = new byte[(temp.length / 4) * 3];
    for (int i = 0; i < temp.length / 4; i++) {
```

```
            pixels[i * 3] = temp[i * 4 + 2];
            pixels[i * 3 + 1] = temp[i * 4 + 1];
            pixels[i * 3 + 2] = temp[i * 4];
        }
        return pixels;
    }
```

2.5 阈值推荐

阈值区间为[0~1]，建议阈值设置为 0.8，可根据实际场景需求进行调整。

3.常见问题

3.1 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: Android 人脸检测结果的人脸框绘制到 View 上为何位置不对？

A: 人脸检测结果的人脸框位置是基于输入图像的，例如在竖屏模式下，假设 View 的宽高是 1080x1920，相机是后置相机，并且预览数据宽高为 1920x1080，有一个被检测到的人脸位置是（left,top,right,bottom），那么需要绘制到 View 上的 Rect 就是（bottom,left,1080-top,right），相当于顺时针旋转 90 度，其他角度可用类似的方法计算。另外，在一般情况下，安卓调用前置相机时在 View 上的显示的画面和实际预览数据成镜像关系。

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: 86018 即需要检测的属性未初始化，需要查看调用接口的属性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。

3.2 其他帮助

可在论坛寻求帮助，SDK 交流论坛：http://ai.arcsoft.com.cn/bbs/