ArcSoft ArcFace SDK

开发说明文档

版本历史

SDK 版本	日期	作者	描述
2.2.12402010101.3	06/20/2019	ArcSoft	优化算法库，新增 IR 活体检测、活体检测阈值设置接口

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、RGB 活体检测、IR 活体检测等能力，初次使用时需联网激活，激活后即可在本地无网络环境下工作，可根据具体的业务需求结合人脸识别 SDK 灵活地进行应用层开发。

1.2 环境要求

1.2.1 系统要求

Linux x64

1.2.2 开发环境

库依赖 GLIBC 2.17 及以上
库依赖 GLIBCXX 3.4.19 及以上
编译器 GCC 4.8.2 及以上

1.2.3 支持的颜色空间格式

常量名	常量值	颜色格式说明
ASVL_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
ASVL_PAF_RGB24_B8G8R8	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
ASVL_PAF_GRAY	1793	8-bit IR 图像
ASVL_PAF_DEPTH_U16	3074	16-bit IR 图像

1.3 产品功能简介

1.3.1 人脸检测

对传入的图像数据进行人脸检测，返回人脸的边框以及朝向信息，可用于后续的人脸识别、活体检测等操作;

支持 image 模式和 video 模式;

支持单人脸、多人脸检测，最多支持检测人脸检测数为 50。

1.3.2 人脸跟踪

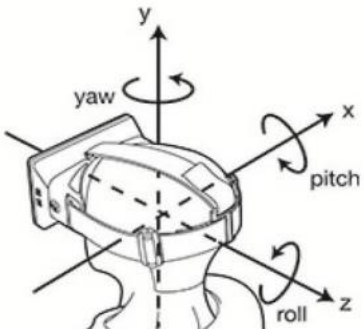
对来自于视频流中的图像数据，进行人脸检测，并对检测到的人脸进行持续跟踪。

1.3.3 人脸属性检测

人脸属性分析，支持性别、年龄等。

1.3.4 人脸三维角度检测

分析人脸的三维角度信息，具体为：俯仰角（pitch），横滚角（roll），偏航角（yaw）。



1.3.5 人脸特征提取

提取人脸视觉特征信息。

1.3.6 人脸比对

对两个人脸特征数据进行比对，来判断是否为同一个人，返回比对相似度值。

1.3.7 活体检测

离线活体检测，静默式识别，在人脸识别过程中判断操作用户是否为真人，有效防御照片、视频、纸张等不同类型的作弊攻击，提高业务安全性，让人脸识别更安全、更快捷，体验更佳。支持单目 RGB 活体检测、双目（IR/RGB）活体检测，可满足各类人脸识别终端产品活体检测应用。

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，此授权的校验是基于设备的唯一标识，被授权的设备在初次授权时需在线进行授权，授权成功后可以离线运行 SDK。

在线授权：

a) 首次激活需保证与公网连通；

b) 调用在线激活接口激活 SDK；

注意事项：

a) 设备授权后，若设备授权信息被删除（重装系统/应用被卸载等），需联网重新激活；

b) 硬件信息发生变更，需要重新激活；

2.SDK 接入指南

2.1 获取 SDK

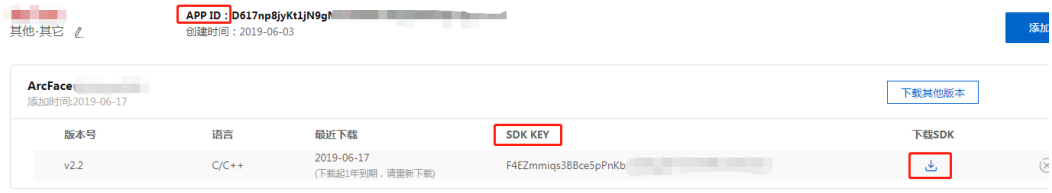
2.1.1 注册开发者账号

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

创建应用，添加 SDK。选择对应信息，确认后即可下载 SDK 和查看对应信息。

查看所需要 APPID、SDKKEY，点击下载图标获取 SDK 开发包。



2.1.3 SDK 包结构

```
|---doc
|    |---ARCSOFT_ARC_FACE_DEVELOPER'S_GUIDE.pdf        开发说明文档
|---inc
|    |---amcomdef.h                                    平台文件
|    |---asvloffscreen.h                               平台文件
|    |---arcsoft_face_sdk.h                            接口文件
|    |---merror.h                                      错误码文件
|---lib
|    |---linux_x64
|    |    |---libarcsoft_face.so                       算法库
|    |    |---libarcsoft_face_engine.so                引擎库
|---samplecode
|    |---ASFTestDemo                                   示例Demo
|    |---ReadMe.txt                                    Demo使用说明
|---releasenotes.txt                                   说明文件
```

2.1.4 工程配置

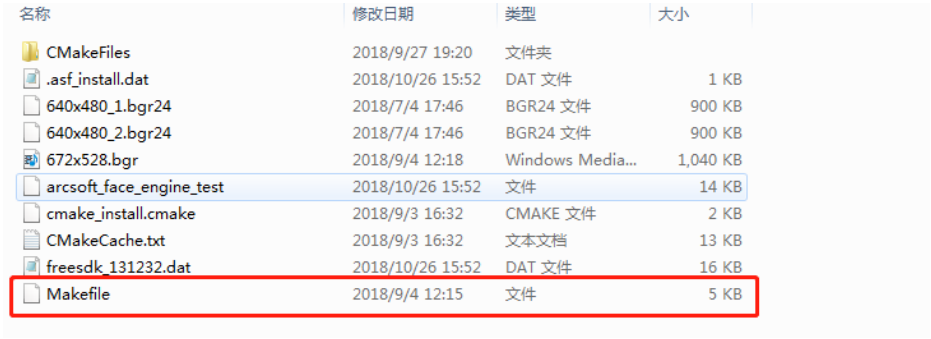
CMakeLists.txt 的步骤：

```
cmake_minimum_required(VERSION 2.0)      #指定 cmake 版本
project(项目名称)                          #指定项目的名称
set(CMAKE_CXX_STANDARD 11)               #设置 c++标准

#指定头文件目录
include_directories(./)
include_directories(./inc)

#指定静态和动态文件目录
link_directories(../linux_so)

add_executable(arcsoft_face_engine_test
    ./inc/amcomdef.h
    ./inc/arcsoft_face_sdk.h
    ./inc/asvloffscreen.h
    ./inc/merror.h
    ./AFCTestDemo.cpp)

#在给定的作用域内设置一个命名的属性
set_property(TARGET arcsoft_face_engine_test
             PROPERTY POSITION_INDEPENDENT_CODE ON)

#链接库文件
target_link_libraries(arcsoft_face_engine_test
    arcsoft_face
    arcsoft_face_engine
)
```

相对路径说明：

CMakeLists.txt 通过 cmake 命令编译生成 makefile 文件为起点，去找工程中文件配置目录。上面的 CMakeLists.txt 编写示例是基于如下工程目录：

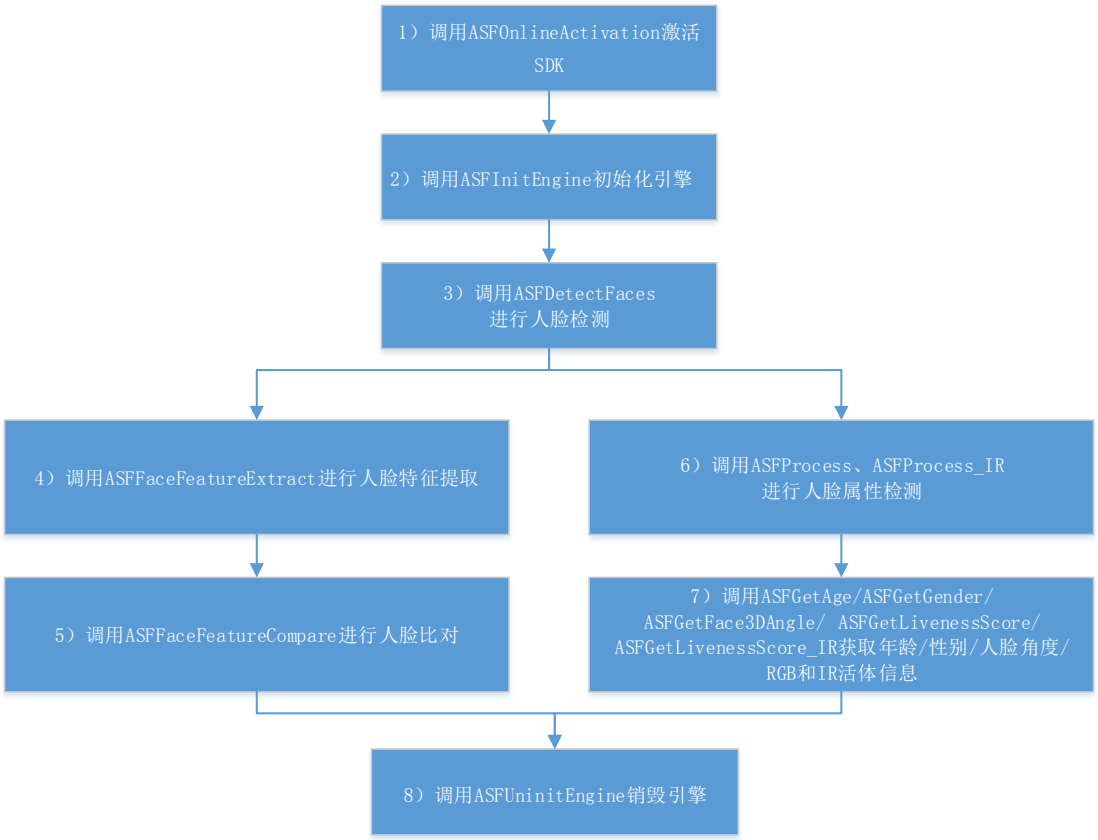
生成的 makefile 文件在 build 文件夹中：



头文件在 inc 文件夹中；

so 文件在 linux_so 文件中；

2.1.5 调用流程图



2.1.6 阈值推荐

a) 人脸比对阈值,相似度区间为[0~1]，推荐阈值为 0.8；

b) RGB 活体检测阈值，设置区间为[0~1]，默认阈值 0.75；

c) IR 活体检测阈值，设置区间为[0~1]，默认阈值 0.7；

以上阈值可根据实际使用场景具体调整。

2.2 数据结构

2.2.1 ASF_VERSION

结构体描述：

SDK 版本信息

定义：

```
typedef struct {
```

```
    MPChar Version;                 // 版本号
    MPChar BuildDate;               // 构建日期
    MPChar CopyRight;               // 版权说明
}ASF_VERSION;
```

2.2.2 ASF_ActiveFileInfo

结构体描述：

激活文件信息

定义：

```
typedef struct{
    MPChar startTime;           //SDK 开始时间
    MPChar endTime;             //SDK 截止时间
    MPChar platform;            //平台版本
    MPChar sdkType;             //SDK 类型
    MPChar appId;               //APPID
    MPChar sdkKey;              //SDKKEY
    MPChar sdkVersion;          //SDK 版本号
    MPChar fileVersion;         //激活文件版本号
}ASF_ActiveFileInfo, *LPASF_ActiveFileInfo;
```

2.2.3 ASF_SingleFaceInfo

结构体描述：

单人脸信息

定义：

```
typedef struct {
    MRECT       faceRect;       // 人脸框
    MInt32      faceOrient;     //人脸角度
} ASF_SingleFaceInfo, *LPASF_SingleFaceInfo;
```

2.2.4 ASF_MultiFaceInfo

结构体描述：

多人脸信息

定义：

```
typedef struct {
    MRECT*      faceRect;           // 人脸框数组
    MInt32*     faceOrient;         // 人脸角度数组
    MInt32      faceNum;            // 检测到的人脸个数
    MInt32*     faceID;             //在 VIDEO 模式下有效，IMAGE 模式下为空
}ASF_MultiFaceInfo, *LPASF_MultiFaceInfo;
```

2.2.5 ASF_FaceFeature

结构体描述：

人脸特征

定义：

```
typedef struct {
    MByte*      feature;        // 人脸特征
    MInt32      featureSize;    // 人脸特征长度
}ASF_FaceFeature, *LPASF_FaceFeature;
```

2.2.6 ASF_AgeInfo

结构体描述：

年龄信息

定义：

```
typedef struct{
    MInt32* ageArray;   // 0:未知; >0:年龄
    MInt32  num;        // 检测的人脸个数
}ASF_AgeInfo, *LPASF_AgeInfo;
```

2.2.7 ASF_GenderInfo

结构体描述：

性别信息

定义：

```
typedef struct{
    MInt32* genderArray;    // 0:男性; 1:女性; -1:未知
    MInt32  num;            // 检测的人脸个数
}ASF_GenderInfo, *LPASF_GenderInfo;
```

2.2.8 ASF_Face3DAngle

结构体描述：

3D 角度信息

定义：

```
typedef struct{
    MFloat* roll;           //横滚角
    MFloat* yaw;            //偏航角
    MFloat* pitch;          //俯仰角
    MInt32* status;         //0:正常; 非 0:异常
    MInt32 num;             //检测的人脸个数
```

```
}ASF_Face3DAngle, *LPASF_Face3DAngle;
```

2.2.9 ASF_LivenessThreshold

结构体描述：

活体置信度

定义：

```
typedef struct{
    MFloat      thresholdmodel_BGR;         //RGB 活体置信度
    MFloat      thresholdmodel_IR;          //IR 活体置信度
}ASF_LivenessThreshold, *LPASF_LivenessThreshold;
```

2.2.10 ASF_LivenessInfo

结构体描述：

活体信息

定义：

```
typedef struct{
        MInt32* isLive;     //0:非真人； 1:真人； -1：不确定； -2:传入人脸数>1
        MInt32 num;         //检测的人脸个数
}ASF_LivenessInfo, *LPASF_LivenessInfo;
```

2.3 枚举

2.3.1 人脸检测方向

根据应用场景，推荐选择单一角度，检测效果更优；

```
enum ArcSoftFace_OrientPriority {
    ASF_OP_0_ONLY = 0x1,        // 仅检测 0 度
    ASF_OP_90_ONLY = 0x2,       // 仅检测 90 度
    ASF_OP_270_ONLY = 0x3,      // 仅检测 270 度
    ASF_OP_180_ONLY = 0x4,      // 仅检测 180 度
    ASF_OP_0_HIGHER_EXT = 0x5,  // 全角度检测
};
```

注：IMAGE 模式下为了提高检测识别率，不支持 ASF_OP_0_HIGHER_EXT 检测

2.3.2 检测到的人脸角度（按逆时针方向）

```
enum ArcSoftFace_OrientCode {
    ASF_OC_0 = 0x1,         // 0 度
    ASF_OC_90 = 0x2,        // 90 度
```

```
    ASF_OC_270 = 0x3,       // 270 度
    ASF_OC_180 = 0x4,       // 180 度
    ASF_OC_30 = 0x5,        // 30 度
    ASF_OC_60 = 0x6,        // 60 度
    ASF_OC_120 = 0x7,       // 120 度
    ASF_OC_150 = 0x8,       // 150 度
    ASF_OC_210 = 0x9,       // 210 度
    ASF_OC_240 = 0xa,       // 240 度
    ASF_OC_300 = 0xb,       // 300 度
    ASF_OC_330 = 0xc        // 330 度
};
```

3.接口

3.1 接口说明

3.1.1 ASFGetActiveFileInfo

接口

```
MRESULT ASFGetActiveFileInfo(
    LPASF_ActiveFileInfo  activeFileInfo
);
```

功能描述

获取激活文件信息

参数

activeFileInfo　[out]　激活文件信息

返回值

成功返回 MOK，失败详见 3.2 错误码列表

3.1.2 ASFOnlineActivation

接口

```
MRESULT ASFOnlineActivation(
    MPChar    AppId,
    MPChar    SDKKey
);
```

功能描述

用于在线激活 SDK

注：

1）初次使用 SDK 时需要对 SDK 先进行激活，激活后无需重复调用；

2）调用此接口时必须为联网状态，激活成功后即可离线使用；

参数

AppId	[in]	官网获取的 APPID
SDKKey	[in]	官网获取的 SDKKEY

返回值

成功返回 MOK 或 MERR_ASF_ALREADY_ACTIVATED，失败详见 3.2 错误码列表

3.1.3 ASFActivation

接口

```
MRESULT ASFActivation(
    MPChar    AppId,
    MPChar    SDKKey
);
```

功能描述

在线激活 SDK，与 ASFOnlineActivation 接口功能相同，首次需要联网激活，生成激活文件之后即可进行本地校验。（ASFOnlineActivation 接口做了激活方案的优化，推荐使用 ASFOnlineActivation 接口）

参数

AppId	[in]	官网获取的 APPID
SDKKey	[in]	官网获取的 SDKKEY

返回值

成功返回 MOK 或 MERR_ASF_ALREADY_ACTIVATED，失败详见 3.2 错误码列表

3.1.4 ASFInitEngine

接口

```
MRESULT ASFInitEngine(
    MUInt32                detectMode,
    ASF_OrientPriority     detectFaceOrientPriority,
    MInt32                 detectFaceScaleVal,
    MInt32                 detectFaceMaxNum,
    MInt32                 combinedMask,
    MHandle*               hEngine
);
```

功能描述

初始化引擎

参数

detectMode	[in]	VIDEO 模式/IMAGE 模式 VIDEO 模式:处理连续帧的图像数据 IMAGE 模式:处理单张的图像数据
detectFaceOrientPriority	[in]	人脸检测角度，推荐单一角度检测；IMAGE 模式下不支持全角度（ASF_OP_0_HIGHER_EXT）检测
detectFaceScaleVal	[in]	识别的最小人脸比例（图片长边与人脸框长边的比值） VIDEO 模式取值范围[2,32]，推荐值为 16 IMAGE 模式取值范围[2,32]，推荐值为 30
detectFaceMaxNum	[in]	最大需要检测的人脸个数，取值范围[1,50]
combinedMask	[in]	需要启用的功能组合，可多选
hEngine	[out]	引擎句柄

返回值

成功返回 MOK，失败详见 3.2 错误码列表

3.1.5 ASFDetectFaces

接口

```
MRESULT ASFDetectFaces(
    MHandle                 hEngine,
    MInt32                  width,
    MInt32                  height,
    MInt32                  format,
    MUInt8*                 imgData,
    LPASF_MultiFaceInfo     detectedFaces
);
```

功能描述

人脸检测

参数

hEngine	[in]	引擎句柄
width	[in]	图片宽度，为 4 的倍数
height	[in]	图片高度，YUYV/I420/NV21/NV12 格式为 2 的倍数； BGR24/GRAY/DEPTH_U16 格式无限制
format	[in]	颜色空间格式
imgData	[in]	图片数据
detectedFaces	[out]	检测到的人脸信息

返回值

成功返回 MOK，失败详见 3.2 错误码列表

3.1.6 ASFFaceFeatureExtract

接口

```
MRESULT ASFFaceFeatureExtract(
    MHandle                 hEngine,
    MInt32                  width,
    MInt32                  height,
    MInt32                  format,
    MUInt8*                 imgData,
    LPASF_SingleFaceInfo    faceInfo,
    LPASF_FaceFeature       feature
);
```

功能描述

单人脸特征提取

参数

hEngine	[in]	引擎句柄
width	[in]	图片宽度，为 4 的倍数
height	[in]	图片高度，YUYV/I420/NV21/NV12 格式为 2 的倍数；BGR24/GRAY 格式无限制
format	[in]	颜色空间格式
imgData	[in]	图片数据
faceInfo	[in]	单张人脸信息
feature	[out]	人脸特征

返回值

成功返回 MOK，失败详见 3.2 错误码列表

3.1.7 ASFFaceFeatureCompare

接口

```
MRESULT ASFFaceFeatureCompare(
    MHandle                 hEngine,
    LPASF_FaceFeature       feature1,
    LPASF_FaceFeature       feature2,
    MFloat*                 confidenceLevel
);
```

功能描述

人脸特征比对

参数

hEngine	[in]	引擎句柄

feature1	[in]	人脸特征值
feature2	[in]	人脸特征值
confidenceLevel	[out]	比对结果，相似度

返回值

成功返回 MOK，失败详见 3.2 错误码列表

3.1.8 ASFSetLivenessParam

接口

```
MRESULT ASFSetLivenessParam(
    MHandle                     hEngine,
    LPASF_LivenessThreshold     threshold
);
```

功能描述

修改 RGB/IR 活体阈值，SDK 默认 RGB：0.75,IR：0.7

参数

hEngine	[in]	引擎句柄
threshold	[in]	活体置信度，推荐阈值 RGB:0.75， IR:0.7

返回值

成功返回 MOK，失败详见 3.2 错误码列表

3.1.9 ASFProcess

接口

```
MRESULT ASFProcess(
    MHandle                 hEngine,
    MInt32                  width,
    MInt32                  height,
    MInt32                  format,
    MUInt8*                 imgData,
    LPASF_MultiFaceInfo     detectedFaces,
    MInt32                  combinedMask
);
```

功能描述

人脸信息检测（年龄/性别/人脸 3D 角度），最多支持 4 张人脸信息检测，超过部分返回未知（活体仅支持单张人脸检测，超出返回未知）,接口不支持 IR 图像检测。

参数

hEngine	[in]	引擎句柄
width	[in]	图片宽度，为 4 的倍数

height	[in]	图片高度，YUYV/I420/NV21/NV12 格式为 2 的倍数，BGR24 格式无限制
format	[in]	颜色空间格式
imgData	[in]	图片数据
detectedFaces	[in]	检测到的人脸信息
combinedMask	[in]	检测的属性（ASF_AGE、ASF_GENDER、ASF_FACE3DANGLE、ASF_LIVENESS），支持多选 注：检测的属性须在引擎初始化接口的 combinedMask 参数中启用

返回值

成功返回 MOK，失败详见 3.2 错误码列表

3.1.10 ASFProcess_IR

接口

```
MRESULT ASFProcess_IR(
    MHandle                 hEngine,
    MInt32                  width,
    MInt32                  height,
    MInt32                  format,
    MUInt8*                 imgData,
    LPASF_MultiFaceInfo     detectedFaces,
    MInt32                  combinedMask
);
```

功能描述

IR 活体单人脸检测

参数

hEngine	[in]	引擎句柄
width	[in]	图片宽度，为 4 的倍数
height	[in]	图片高度，I420/NV21/NV12 格式为 2 的倍数，DEPTH_U16/GRAY 格式无限制
format	[in]	颜色空间格式， 支持（I420/NV21/NV12/DEPTH_U16/GRAY）的检测
imgData	[in]	图片数据
detectedFaces	[in]	人脸信息，用户根据待检测的功能选择需要使用的人脸。
combinedMask	[in]	检测的属性（ASF_IR_LIVENESS） 注：检测的属性须在引擎初始化接口的 combinedMask 参数中启用

返回值

成功返回 MOK，失败详见 3.2 错误码列表

3.1.11 ASFGetAge

接口

```
MRESULT ASFGetAge(
    MHandle             hEngine,
    LPASF_AgeInfo       ageInfo
);
```

功能描述

获取年龄信息

参数

hEngine	[in]	引擎句柄
ageInfo	[out]	检测到的年龄信息

返回值

成功返回 MOK，失败详见 3.2 错误码列表

3.1.12 ASFGetGender

接口

```
MRESULT ASFGetGender(
    MHandle                 hEngine,
    LPASF_GenderInfo        genderInfo
);
```

功能描述

获取性别信息

参数

hEngine	[in]	引擎句柄
genderInfo	[out]	检测到的性别信息

返回值

成功返回 MOK，失败详见 3.2 错误码列表

3.1.13 ASFGetFace3DAngle

接口

```
MRESULT ASFGetFace3DAngle(
    MHandle                 hEngine,
    LPASF_Face3DAngle       p3DAngleInfo
);
```

功能描述

获取 3D 角度信息

参数

hEngine	[in]	引擎句柄
p3DAngleInfo	[out]	检测到脸部 3D 角度信息

返回值

成功返回 MOK，失败详见 3.2 错误码列表

3.1.14 ASFGetLivenessScore

接口

```
MRESULT ASFGetLivenessScore(
    MHandle              hEngine,
    LPASF_LivenessInfo   livenessInfo
);
```

功能描述

获取 RGB 活体信息

参数

hEngine	[in]	引擎句柄
livenessInfo	[out]	RGB 活体信息，详见 2.2.10 ASF_LivenessInfo

返回值

成功返回 MOK，失败详见 3.2 错误码列表

3.1.15 ASFGetLivenessScore_IR

接口

```
MRESULT ASFGetLivenessScore_IR(
    MHandle              hEngine,
    LPASF_LivenessInfo   livenessInfo
);
```

功能描述

获取 IR 活体信息

参数

hEngine	[in]	引擎句柄
livenessInfo	[out]	IR 活体信息，详见 2.2.10 ASF_LivenessInfo

返回值

成功返回 MOK，失败详见 3.2 错误码列表

3.1.16 ASFGetVersion

接口

```
const ASF_VERSION* ASFGetVersion(
    MHandle    hEngine
);
```

功能描述

获取版本信息

参数

hEngine	[in]	引擎句柄

返回值

成功返回版本信息，失败返回 MNull。

3.1.17 ASFUninitEngine

接口

```
MRESULT ASFUninitEngine(
    MHandle hEngine
);
```

功能描述

销毁引擎

参数

hEngine	[in]	引擎句柄

返回值

成功返回 MOK，失败详见 3.2 错误码列表

3.2 错误码列表

错误码名	十六进制	十进制	错误码说明
MOK	0x0	0	成功
MERR_UNKNOWN	0x1	1	错误原因不明
MERR_INVALID_PARAM	0x2	2	无效的参数
MERR_UNSUPPORTED	0x3	3	引擎不支持
MERR_NO_MEMORY	0x4	4	内存不足
MERR_BAD_STATE	0x5	5	状态错误
MERR_USER_CANCEL	0x6	6	用户取消相关操作
MERR_EXPIRED	0x7	7	操作时间过期

MERR_USER_PAUSE	0x8	8	用户暂停操作
MERR_BUFFER_OVERFLOW	0x9	9	缓冲上溢
MERR_BUFFER_UNDERFLOW	0xA	10	缓冲下溢
MERR_NO_DISKSPACE	0xB	11	存贮空间不足
MERR_COMPONENT_NOT_EXIST	0xC	12	组件不存在
MERR_GLOBAL_DATA_NOT_EXIST	0xD	13	全局数据不存在
MERR_FSDK_INVALID_APP_ID	0x7001	28673	无效的 AppId
MERR_FSDK_INVALID_SDK_ID	0x7002	28674	无效的 SDKkey
MERR_FSDK_INVALID_ID_PAIR	0x7003	28675	AppId 和 SDKKey 不匹配
MERR_FSDK_MISMATCH_ID_AND_SDK	0x7004	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	0x7005	28677	系统版本不被当前 SDK 所支持
MERR_FSDK_LICENCE_EXPIRED	0x7006	28678	SDK 有效期过期，需要重新下载更新
MERR_FSDK_FR_INVALID_MEMORY_INFO	0x12001	73729	无效的输入内存
MERR_FSDK_FR_INVALID_IMAGE_INFO	0x12002	73730	无效的输入图像参数
MERR_FSDK_FR_INVALID_FACE_INFO	0x12003	73731	无效的脸部信息
MERR_FSDK_FR_MISMATCHED_FEATURE_LEVEL	0x12005	73733	待比较的两个人脸特征的版本不一致
MERR_FSDK_FACEFEATURE_UNKNOWN	0x14001	81921	人脸特征检测错误未知
MERR_FSDK_FACEFEATURE_MEMORY	0x14002	81922	人脸特征检测内存错误
MERR_FSDK_FACEFEATURE_INVALID_FORMAT	0x14003	81923	人脸特征检测格式错误
MERR_FSDK_FACEFEATURE_INVALID_PARAM	0x14004	81924	人脸特征检测参数错误
MERR_FSDK_FACEFEATURE_LOW_CONFIDENCE_LEVEL	0x14005	81925	人脸特征检测结果置信度低
MERR_ASF_EX_FEATURE_UNSUPPORTED_ON_INIT	0x15001	86017	Engine 不支持的检测属性
MERR_ASF_EX_FEATURE_UNINITED	0x15002	86018	需要检测的属性未初始化
MERR_ASF_EX_FEATURE_UNPROCESSED	0x15003	86019	待获取的属性未在 process 中处理过
MERR_ASF_EX_FEATURE_UNSUPPORTED_ON_PROCESS	0x15004	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	0x15005	86021	无效的输入图像

MERR_ASF_EX_INVALID_FACE_INFO	0x15006	86022	无效的脸部信息
MERR_ASF_ACTIVATION_FAIL	0x16001	90113	SDK 激活失败，请打开读写权限
MERR_ASF_ALREADY_ACTIVATED	0x16002	90114	SDK 已激活
MERR_ASF_NOT_ACTIVATED	0x16003	90115	SDK 未激活
MERR_ASF_SCALE_NOT_SUPPORT	0x16004	90116	detectFaceScaleVal 不支持
MERR_ASF_ACTIVEFILE_SDKTYPE_MISMATCH	0x16005	90117	激活文件与 SDK 类型不匹配,请确认使用的 sdk
MERR_ASF_DEVICE_MISMATCH	0x16006	90118	设备不匹配
MERR_ASF_UNIQUE_IDENTIFIER_ILLEGAL	0x16007	90119	唯一标识不合法
MERR_ASF_PARAM_NULL	0x16008	90120	参数为空
MERR_ASF_VERSION_NOT_SUPPORT	0x1600A	90122	版本不支持
MERR_ASF_SIGN_ERROR	0x1600B	90123	签名错误
MERR_ASF_DATABASE_ERROR	0x1600C	90124	激活信息保存异常
MERR_ASF_UNIQUE_CHECKOUT_FAIL	0x1600D	90125	唯一标识符校验失败
MERR_ASF_COLOR_SPACE_NOT_SUPPORT	0x1600E	90126	颜色空间不支持
MERR_ASF_IMAGE_WIDTH_HEIGHT_NOT_SUPPORT	0x1600F	90127	图片宽高不支持,宽度需四字节对齐
MERR_ASF_ACTIVATION_DATA_DESTROYED	0x16011	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	0x16012	90130	服务端未知错误
MERR_ASF_ACTIVEFILE_SDK_MISMATCH	0x16014	90132	激活文件与 SDK 版本不匹配,请重新激活
MERR_ASF_DEVICEINFO_LESS	0x16015	90133	设备信息太少,不足以生成设备指纹
MERR_ASF_REQUEST_TIMEOUT	0x16016	90134	客户端时间与服务器时间(即北京时间）前后相差在 30 分钟以上
MERR_ASF_APPID_DATA_DECRYPT	0x16017	90135	数据校验异常
MERR_ASF_APPID_APPKEY_SDK_MISMATCH	0x16018	90136	传入的 AppId 和 AppKey 与使用的 SDK 版本不一致
MERR_ASF_NO_REQUEST	0x16019	90137	短时间大量请求会被禁止请求,30 分钟之后解封
MERR_ASF_ACTIVE_FILE_NO_EXIST	0x1601A	90138	激活文件不存在
MERR_ASF_IMAGEMODE_0_HIGHER_EXT_UNSUPPORT	0x1601B	90139	IMAGE 模式下不支持全角度(ASF_OP_0_HIGHER_EXT)检测
MERR_ASF_NETWORK_COULDNT_RESOLVE_HOST	0x17001	94209	无法解析主机地址
MERR_ASF_NETWORK_COULDNT_CON	0x17002	94210	无法连接服务器

NECT_SERVER			
MERR_ASF_NETWORK_CONNECT_TIMEOUT	0x17003	94211	网络连接超时
MERR_ASF_NETWORK_UNKNOWN_ERROR	0x17004	94212	网络未知错误

3.3 示例代码

```
#include "arcsoft_face_sdk.h"
#include "amcomdef.h"
#include "asvloffscreen.h"
#include "merror.h"
#include <iostream>
#include <string>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define APPID "官网获取APPID"
#define SDKKey "官网获取SDKKEY"

#define SafeFree(p) { if ((p)) free(p); (p) = NULL; }
#define SafeArrayDelete(p) { if ((p)) delete [] (p); (p) = NULL; }
#define SafeDelete(p) { if ((p)) delete (p); (p) = NULL; }

#define NSCALE "取值范围[2,32]，VIDEO模式推荐值16，IMAGE模式内部设置固定值为30"
#define FACENUM  "检测的人脸数"

int main()
{
    //激活SDK
    MRESULT res = ASFOnlineActivation(APPID, SDKKEY);
    if (MOK != res && MERR_ASF_ALREADY_ACTIVATED != res)
        printf("ASFOnlineActivation fail: %d\n", res);
    else
        printf("ASFOnlineActivation sucess: %d\n", res);

    //初始化引擎
    MHandle handle = NULL;
    MInt32 mask = ASF_FACE_DETECT | ASF_FACERECOGNITION | ASF_AGE | ASF_GENDER |
ASF_FACE3DANGLE | ASF_LIVENESS | ASF_IR_LIVENESS;
    res = ASFInitEngine(ASF_DETECT_MODE_IMAGE, ASF_OP_0_ONLY, NSCALE, FACENUM, mask,
&handle);
    if (res != MOK)
```

```
        printf("ALInitEngine fail: %d\n", res);
    else
        printf("ALInitEngine sucess: %d\n", res);

    char* picPath1 = "图片路径，该demo只支持裸数据（NV21/BGR24/BMP等）格式";
    int Width1 = 图片宽度;
    int Height1 = 图片高度;
    int Format1 = ASVL_PAF_RGB24_B8G8R8;    //图像数据为RGB24颜色格式
    MUInt8* imageData1 = (MUInt8*)malloc(Height1*Width1*3);
    FILE* fp1 = fopen(picPath1, "rb");

    char* picPath2 = "图片路径，该demo只支持裸数据（NV21/BGR24/BMP等）格式";
    int Width2 = 图片宽度;
    int Height2 = 图片高度;
    int Format2 = ASVL_PAF_RGB24_B8G8R8;    //图像数据为RGB24颜色格式
    MUInt8* imageData2 = (MUInt8*)malloc(Height2*Width2*3);
    FILE* fp2 = fopen(picPath2, "rb");

    char* picPath3 = "图片路径，这里用的NV21为例";
    int Width3 = 图片宽度;
    int Height3 = 图片高度;
    int Format3 = ASVL_PAF_GRAY;   //用于红外活体检测
    //只读NV21前2/3的数据为灰度数据
    MUInt8* imageData3 = (MUInt8*)malloc(Height2*Width2);
    FILE* fp3 = fopen(picPath3, "rb");

    if (fp1 && fp2 && fp3)
    {
        fread(imageData1, 1, Height1*Width1*3, fp1);     //读BGR裸数据
        fclose(fp1);
        fread(imageData2, 1, Height2*Width2*3, fp2);     //读BGR裸数据
        fclose(fp2);
        fread(imageData3, 1, Height3*Width3, fp3);//读NV21前2/3的数据,用于红外活体检测
        fclose(fp3);

        // 人脸检测
        ASF_MultiFaceInfo detectedFaces1 = { 0 };
        ASF_SingleFaceInfo SingleDetectedFaces = { 0 };
        ASF_FaceFeature feature1 = { 0 };
        ASF_FaceFeature copyfeature1 = { 0 };
        res = ASFDetectFaces(handle, Width1, Height1, Format1, imageData1,
&detectedFaces1);
        if (res != MOK)
            printf("%s ASFDetectFaces fail: %d\n", picPath1, res);
```

```
        else
        {
            printf("%s ASFDetectFaces sucess: %d\n", picPath1, res);
            SingleDetectedFaces.faceRect.left = detectedFaces1.faceRect[0].left;
            SingleDetectedFaces.faceRect.top = detectedFaces1.faceRect[0].top;
            SingleDetectedFaces.faceRect.right = detectedFaces1.faceRect[0].right;
            SingleDetectedFaces.faceRect.bottom = detectedFaces1.faceRect[0].bottom;
            SingleDetectedFaces.faceOrient = detectedFaces1.faceOrient[0];
        }

        // 单人脸特征提取
        res = ASFFaceFeatureExtract(handle, Width1, Height1, Format1, imageData1,
&SingleDetectedFaces, &feature1);
        if (res != MOK)
            printf("%s ASFFaceFeatureExtract fail: %d\n", picPath1, res);
        else
        {
            printf("%s ASFFaceFeatureExtract sucess: %d\n", picPath1, res);
            //拷贝feature，否则第二次进行特征提取，会覆盖第一次特征提取的数据，导致比对
的结果为1
            copyfeature1.featureSize = feature1.featureSize;
            copyfeature1.feature = (MByte *)malloc(feature1.featureSize);
            memset(copyfeature1.feature, 0, feature1.featureSize);
            memcpy(copyfeature1.feature, feature1.feature, feature1.featureSize);
        }

        ASF_MultiFaceInfo detectedFaces2 = { 0 };
        ASF_FaceFeature feature2 = { 0 };
        res = ASFDetectFaces(handle, Width2, Height2, Format2, imageData2,
&detectedFaces2);
        if (res != MOK)
            printf("%s ASFDetectFaces fail: %d\n", picPath2, res);
        else
        {
            printf("%s ASFDetectFaces sucess: %d\n", picPath2, res);
            SingleDetectedFaces.faceRect.left = detectedFaces2.faceRect[0].left;
            SingleDetectedFaces.faceRect.top = detectedFaces2.faceRect[0].top;
            SingleDetectedFaces.faceRect.right = detectedFaces2.faceRect[0].right;
            SingleDetectedFaces.faceRect.bottom = detectedFaces2.faceRect[0].bottom;
            SingleDetectedFaces.faceOrient = detectedFaces2.faceOrient[0];
        }

        res = ASFFaceFeatureExtract(handle, Width2, Height2, Format2, imageData2,
&SingleDetectedFaces, &feature2);
```

```
        if (res != MOK)
            printf("%s ASFFaceFeatureExtract fail: %d\n", picPath2, res);
        else
            printf("%s ASFFaceFeatureExtract sucess: %d\n", picPath2, res);

        // 单人脸特征比对
        MFloat confidenceLevel;
        res = ASFFaceFeatureCompare(handle, &copyfeature1, &feature2, &confidenceLevel);
        if (res != MOK)
            printf("ASFFaceFeatureCompare fail: %d\n", res);
        else
            printf("ASFFaceFeatureCompare sucess: %lf\n", confidenceLevel);

        //设置活体置信度 SDK内部默认值为 IR：0.7  RGB：0.75（无特殊需要，可以不设置）
        ASF_LivenessThreshold threshold = { 0 };
        threshold.thresholdmodel_BGR = 0.75;
        threshold.thresholdmodel_IR = 0.7;
        res = ASFSetLivenessParam(handle, &threshold);
        if (res != MOK)
            printf("ASFSetLivenessParam fail: %d\n", res);
        else
            printf("ASFSetLivenessParam sucess: %d\n", res);

        // 人脸信息检测
        MInt32 lastMask = ASF_AGE | ASF_GENDER | ASF_FACE3DANGLE | ASF_LIVENESS;
        res = ASFProcess(handle, Width2, Height2, Format2, imageData2, &detectedFaces2,
lastMask);
        if (res != MOK)
            printf("ASFProcess fail: %d\n", res);
        else
            printf("ASFProcess sucess: %d\n", res);

        // 获取年龄
        ASF_AgeInfo ageInfo = { 0 };
        res = ASFGetAge(handle, &ageInfo);
        if (res != MOK)
            printf("%s ASFGetAge fail: %d\n", picPath2, res);
        else
            printf("%s ASFGetAge sucess: %d First face age: %d\n", picPath2, res,
ageInfo.ageArray[0]);

        // 获取性别
        ASF_GenderInfo genderInfo = { 0 };
        res = ASFGetGender(handle, &genderInfo);
```

```
        if (res != MOK)
            printf("%s ASFGetGender fail: %d\n", picPath2, res);
        else
            printf("%s ASFGetGender sucess: %d First face gender: %d\n", picPath2, res,
genderInfo.genderArray[0]);

        // 获取3D角度
        ASF_Face3DAngle angleInfo = { 0 };
        res = ASFGetFace3DAngle(handle, &angleInfo);
        if (res != MOK)
            printf("%s ASFGetFace3DAngle fail: %d\n", picPath2, res);
        else
            printf("%s ASFGetFace3DAngle sucess: %d First face 3dAngle: roll: %lf yaw: %lf
pitch: %lf\n", picPath2, res, angleInfo.roll[0], angleInfo.yaw[0], angleInfo.pitch[0]);

        //获取活体信息
        ASF_LivenessInfo rgbLivenessInfo = { 0 };
        res = ASFGetLivenessScore(handle, &rgbLivenessInfo);
        if (res != MOK)
            printf("ASFGetLivenessScore fail: %d\n", res);
        else
            printf("ASFGetLivenessScore sucess: %d\n", rgbLivenessInfo.isLive[0]);


        //**************进行IR活体检测********************
        printf("\n**********IR LIVENESS*************\n");

        ASF_MultiFaceInfo detectedFaces3 = { 0 };
        //以GRAY图像为例进行红外活体检测
        res = ASFDetectFaces(handle, Width3, Height3, Format3, imageData3,
&detectedFaces3);
        if (res != MOK)
            printf("ASFDetectFaces fail: %d\n", res);
        else
            printf("Face num: %d\n", detectedFaces3.faceNum);

        //IR图像活体检测
        MInt32 processIRMask = ASF_IR_LIVENESS;
        res = ASFProcess_IR(handle, Width3, Height3, ASVL_PAF_GRAY, imageData3,
&detectedFaces3, processIRMask);
        if (res != MOK)
            printf("ASFProcess_IR fail: %d\n", res);
        else
            printf("ASFProcess_IR sucess: %d\n", res);
```

```
        //获取IR活体信息
        ASF_LivenessInfo irLivenessInfo = { 0 };
        res = ASFGetLivenessScore_IR(handle, &irLivenessInfo);
        if (res != MOK)
            printf("ASFGetLivenessScore_IR fail: %d\n", res);
        else
            printf("IR Liveness: %d\n", irLivenessInfo.isLive[0]);


        SafeFree(copyfeature1.feature);          //释放内存
        SafeArrayDelete(imageData1);
        SafeArrayDelete(imageData2);
        SafeArrayDelete(imageData3);

        //获取版本信息
        const ASF_VERSION* pVersionInfo = ASFGetVersion(handle);

        //反初始化
        res = ASFUninitEngine(handle);
        if (res != MOK)
            printf("ALUninitEngine fail: %d\n", res);
        else
            printf("ALUninitEngine sucess: %d\n", res);
    }
    else{
        printf("No pictures found.\n");
    }

    getchar();
    return 0;
}
```

4.常见问题

4.1 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: 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，对于 video 模式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。image 模式下为提高识别率不支持全角度检测。

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[2,32]，推荐值为 16 ；IMAGE 模式有效值范围[2,32]，推荐值为 30，特殊情况下可根据具体场景进行设置。

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值有没有在初始化引擎时在 combinedMask 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：ArcFace SDK 对图像尺寸做了限制，宽度为 4 的倍数，YUYV/I420/NV21/NV12 格式的图片高度为 2 的倍数，BGR24/GRAY/U16 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: 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。若是使用双目摄像头，则很有可能是两者成像差距很大或两者画面成镜像或旋转的关系。

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。

Q: 如何进行 IR 活体检测？

A：推荐的方案采用双目（RGB/ IR）摄像头，RGB 摄像头数据用于人脸检测，将人脸检测的结果用于 IR 活体检测。需要注意的是，IR 活体检测不支持 BGR24 和 YUYV 颜色空间的图像数据。

Q: 初始化引擎时，传入检测模式为 Image 模式，检测方向为全角度，为何会创建失败？

A：为了提高人脸检测识别率，该版本的 IMAGE 模式不支持全角度检测，开发者需要根据图像中的人脸方向确定人脸检测的角度。

Q: Linux 版本在进行 Image 模式人脸检测时，回传的 faceId 数组为何为空？

A:faceId 属性是 video 模式下的特有属性，在一个人脸进入画面到离开，该值不会改变，image 模式下不支持 faceId。

Q: 进行人脸比对时一般会调用 ASFDetectFaces 和 ASFFaceFeatureExtract 两次，可能会导致比对的相似度一直为 1？

A：初始化引擎之后会提前分配好需要使用的内存，所以第二次调用 ASFDetectFaces 和 ASFFaceFeatureExtract 接口输出的结果会覆盖第一次输出的结果，此时应用层定义的指针指向同一块内存，得到的数据是一样的，所以导致比对结果为 1。在有需要的情况下需要对第一次的算法结果进行深拷贝保存。

4.2 其他帮助

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