



ArcSoft ArcFace SDK

开发说明文档

目录

目录	2
1. 简介	4
1.1 产品概述	4
1.2 环境要求	4
1.2.1 系统要求	4
1.2.2 开发环境	4
1.2.3 支持的颜色空间格式	4
1.3 产品功能简介	4
1.3.1 人脸检测	4
1.3.2 人脸跟踪	5
1.3.3 人脸属性检测	5
1.3.4 人脸三维角度检测	5
1.3.5 人脸比对	5
1.3.6 活体检测	5
1.4 SDK 授权说明	6
2. 接入指南	6
2.1 引擎获取	6
2.1.1 注册为开发者	6
2.1.2 SDK 下载	6
2.1.3 SDK 包结构	7
2.1.4 工程配置	7
2.1.5 调用流程	9
2.1.6 阈值推荐	10
2.2 数据结构	10
2.2.1 ASF_VERSION	10
2.2.2 ASF_SingleFaceInfo	10
2.2.3 ASF_MultiFaceInfo	11
2.2.4 ASF_FaceFeature	11
2.2.5 ASF_AgeInfo	11
2.2.6 ASF_GenderInfo	11
2.2.7 ASF_Face3DAngle	12
2.2.8 ASF_LivenessInfo	12
2.3 枚举	12
2.3.1 检测方向的优先级	12
2.3.2 检测到的人脸角度（按逆时针方向）	13
3. 接口	13
3.1 接口说明	13
3.1.1 ASFActivation	13
3.1.2 ASFInitEngine	14
3.1.3 ASFDetectFaces	14
3.1.4 ASFFaceFeatureExtract	15

3.1.5	ASFFaceFeatureCompare	16
3.1.6	ASFProcess	16
3.1.7	ASFGetAge	17
3.1.8	ASFGetGender	17
3.1.9	ASFGetFace3DAngle	18
3.1.10	ASFGetLivenessScore.....	18
3.1.11	ASFGetVersion	18
3.1.12	ASFUninitEngine	19
3.2	错误码概览.....	19
3.3	示例代码.....	22
4.	常见问题.....	26
4.1	FAQ.....	26
4.2	其他帮助.....	28

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 灵活的进行应用层开发。

1.2 环境要求

1.2.1 系统要求

Win7 及以上

1.2.2 开发环境

VS2013 及以上

1.2.3 支持的颜色空间格式

支持图像的颜色空间格式: NV21,NV12, BGR24,I420,YUYV

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

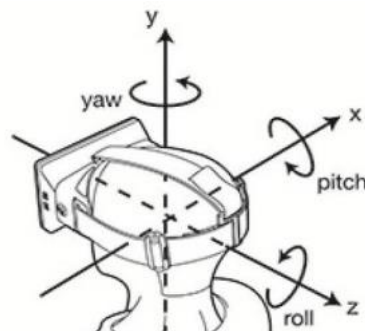
捕捉视频流中的人脸信息，并对人脸进行跟踪。

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，此授权的校验是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，授权成功后可以离线运行 SDK。

激活一台设备后，遇以下情况，设备授权不变，但需要重新联网激活：

- 删除基于 SDK 开发的应用或删除应用数据
- 重新安装系统
- 激活一台设备后，硬件信息发生变更

2.接入指南

2.1 引擎获取

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、对应平台以及版本，确认后即可下载 SDK 和查看激活码。



The screenshot shows a web form titled "ArcFace" for downloading the SDK. It contains four dropdown menus for selection: "选择平台:" (Select Platform) with "Windows(X64)" selected, "选择版本:" (Select Version) with "v2.0" selected, "选择语言:" (Select Language) with "C/C++" selected, and "选择应用:" (Select Application) with "ARCFACE2.0" selected. Below these is a checkbox labeled "我已阅读并同意《虹软公司（ArcSoft）人工智能开放平台服务协议》" (I have read and agree to the ArcSoft AI Open Platform Service Agreement), which is checked. At the bottom are two buttons: "确认" (Confirm) in blue and "取消" (Cancel) in red.

可在查看激活码链接中获取 APPID、SDKKey，点击下载 SDK 下载 SDK 包。

ArcFace v2.0

Windows(X64)

C/C++

免费SDK

添加时间：2019-03-08

[查看激活码](#) | [下载SDK](#) | [删除](#)

有效时间：2020-03-08

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；

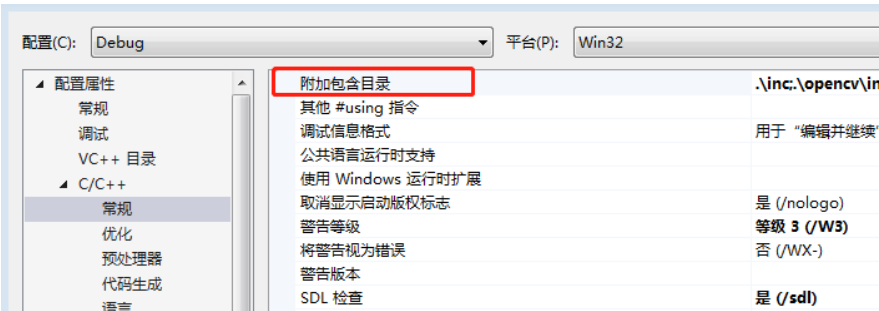
2.1.3 SDK 包结构

---doc	
---ARCSOFT_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	
--- ---X64	
--- ---libarcsoft_face. dll	算法库
--- ---libarcsoft_face_engine. dll	引擎库
--- ---libarcsoft_face_engine. lib	引擎库
---samplecode	
---samplecode. txt	示例代码
---releasenotes. txt	说明文件

2.1.4 工程配置

方法 1：VS 中添加 lib 库及头文件（外部依赖项）的步骤：

- 1、添加工程的头文件目录：
 - a) 右键单击工程名，选择属性---配置属性---c/c+---常规---附加包含目录
 - b) 添加头文件存放目录路径

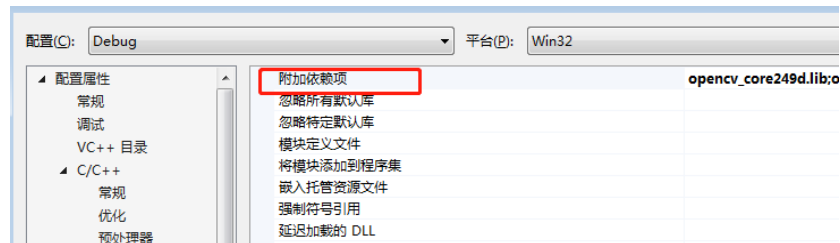


- 2、添加文件引用的 lib 静态库路径：

- a) 右键单击工程名，选择属性---配置属性---链接器---常规---附加库目录
- b) 添加 lib 文件存放目录

3、添加工程引用的 lib 库：

- a) 右键单击工程名，选择属性---配置属性---链接器---输入---附加依赖项
- b) 添加依赖的 lib 库名称



4、添加工程引用的 dll 动态库：

- a) 把引用的 dll 放到工程的可执行文件所在的目录下（例如：debug、release 或工程目录下）。

方法 2：相对路径的设置

在 VS 的工程中常常要设置头文件的包含路径和库目录，如果使用绝对路径，其他人拷贝你的工程到其他机器上就可能无法运行，原因是在建工程时可能把工程放在了 E:盘，但是拷贝后的工程路径和拷贝前的工程路径不一致，会导致找不到头文件问题。因此，建议配置包含路径和库目录时使用相对路径。

这里的相对路径就是相当于工程文件（XXXX.vcproj）为起点计算出的能找到包含所需头文件（也就是找包含所需头文件的 include 目录）的路径。

例如你的工程文件（Count.vcproj）所在目录路径为：

E:\projects\Count\Count\Count.vcproj

该工程需要包含一个图片参数，该图片所在路径如下：

E:\projects\Count\pic\pic01.jpg

这里程序中的相对路径设置如下：

..\pic\pic02.jpg

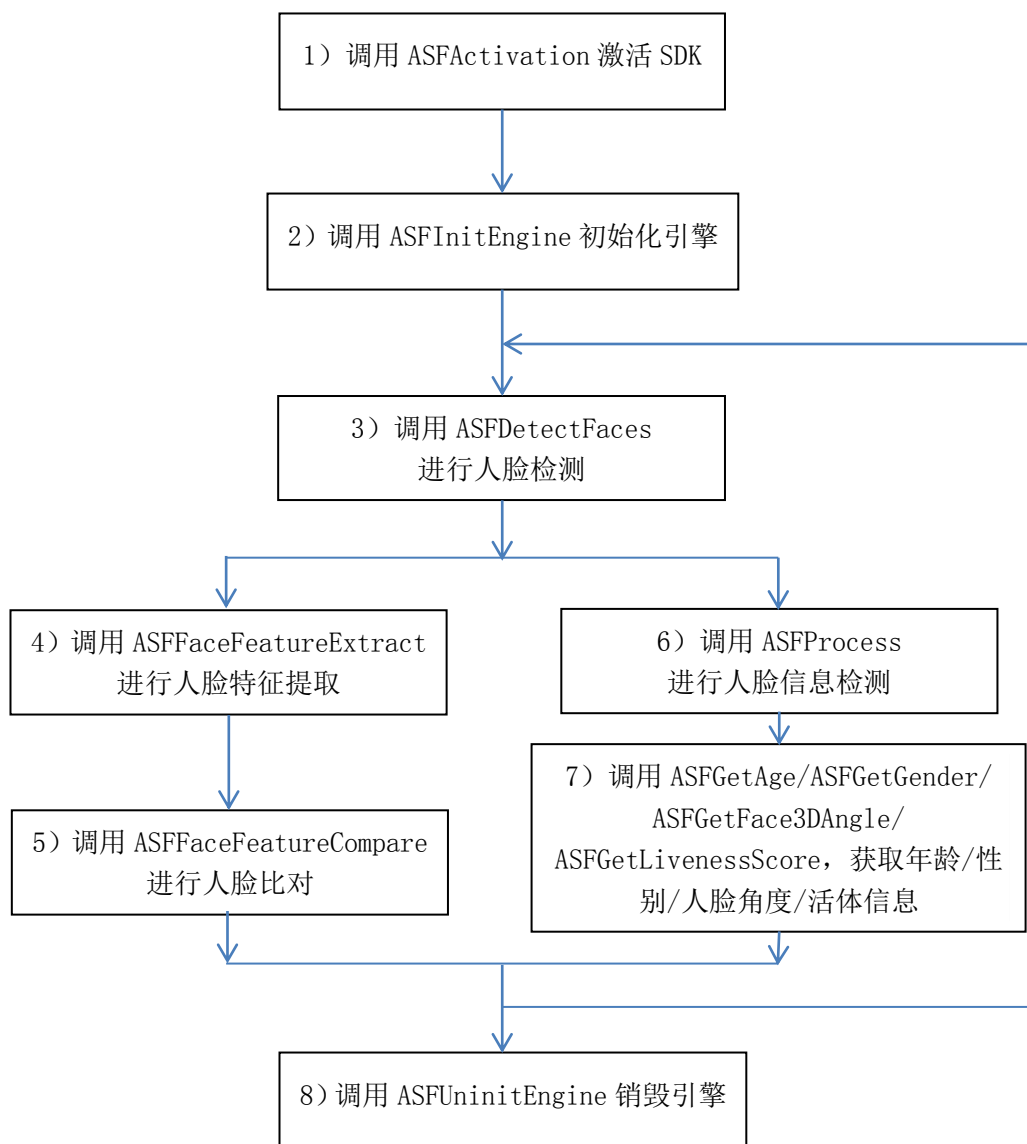
程序代码中的参数路径设置时要用双斜线：

例如：

```
#include "..\\TestLib\\lib.h"
```

```
#pragma comment(lib, "..\\debug\\TestLib.lib");
```


2.1.5 调用流程



Step 1: 激活，调用 ASFActivation

接口所需 AppId、SDKKey 在申请 SDK 时获取，只需在第一次使用时调用激活成功即可；

Step 2: 初始化，调用 ASFINitEngine 初始化引擎

- VIDEO 模式:处理连续帧的图像数据，并返回检测结果，需要将所有图像帧的数据都传入接口进行处理；
- IMAGE 模式:处理单帧的图像数据，并返回检测结果；

Step 3: 调用 ASFDetectFaces 进行人脸检测

接口所需的图像信息，format 参数支持 NV21/NV12/YUYV/BGR24/I420 五种颜色空间格

式，图像处理结果可从 detectedFaces 参数中获取；

Step 4: 调用 ASFFaceFeatureExtract 接口进行人脸特征提取

接口只支持单人脸特征提取，处理结果可从 feature 参数中获取；

Step 5: 调用 ASFFaceFeatureCompare 接口进行人脸比对

接口只支持单人脸比对，处理结果可从 confidenceLevel 参数中获取；

Step 6: 调用 ASFProcess 接口进行人脸信息检测

接口中 combinedMask 参数传入只能是初始化中参数 combinedMask 与 ASF_AGE | ASF_GENDER | ASF_FACE3DANGLE | ASF_LIVENESS 的交集的子集；

Step 7: 调用 ASFGetAge、ASFGetGender、ASFGetFace3Dangle、ASFGetLivenessScore 接口，年龄、性别、人脸角度、活体信息；

Step 8: 调用 ASFUninitEngine 销毁引擎

2.1.6 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。

2.2 数据结构

2.2.1 ASF_VERSION

功能描述：

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_SingleFaceInfo

类描述：

单人脸信息；

定义：

```
typedef struct {
```

```

        MRECT        faceRect;        // 人脸框
        MInt32        faceOrient;      //人脸角度
    } ASF_SingleFaceInfo, *LPASF_SingleFaceInfo;

```

2.2.3 ASF_MultiFaceInfo

类描述:

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
} ASF_MultiFaceInfo, *LPASF_MultiFaceInfo;

```

2.2.4 ASF_FaceFeature

功能描述:

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;

```

2.2.5 ASF_AgeInfo

功能描述:

年龄信息;

定义:

```

typedef struct{
    MInt32* ageArray;    // 0 代表未知, 大于 0 的数值即检测的年龄结果
    MInt32 num;          // 检测的人脸个数
} ASF_AgeInfo, *LPASF_AgeInfo;

```

2.2.6 ASF_GenderInfo

功能描述:

性别信息;

定义:

```

typedef struct{
    MInt32* genderArray;    // 0 表示男性, 1 表示女性, -1 表示未知

```

```

        MInt32 num; // 检测的人脸个数
    }ASF_GenderInfo, *LPASF_GenderInfo;

```

2.2.7 ASF_Face3DAngle

功能描述：

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，其他数值：出错
    MInt32 num; //检测的人脸个数
}ASF_Face3DAngle, *LPASF_Face3DAngle;

```

2.2.8 ASF_LivenessInfo

功能描述：

活体信息；

定义：

```

typedef struct{
    MInt32* isLive; // [out] 判断是否真人, 0: 非真人;
                    // 1: 真人;
                    // -1: 不确定;
                    // -2:传入人脸数>1;

    MInt32 num;
}ASF_LivenessInfo, *LPASF_LivenessInfo;

```

2.3 枚举

2.3.1 检测方向的优先级

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, // 检测 0, 90, 270, 180 全角度
};

```

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 degree
    ASF_OC_90 = 0x2,         // 90 degree
    ASF_OC_270 = 0x3,        // 270 degree
    ASF_OC_180 = 0x4,        // 180 degree
    ASF_OC_30 = 0x5,         // 30 degree
    ASF_OC_60 = 0x6,         // 60 degree
    ASF_OC_120 = 0x7,        // 120 degree
    ASF_OC_150 = 0x8,        // 150 degree
    ASF_OC_210 = 0x9,        // 210 degree
    ASF_OC_240 = 0xa,        // 240 degree
    ASF_OC_300 = 0xb,        // 300 degree
    ASF_OC_330 = 0xc         // 330 degree
};
```

3.接口

3.1 接口说明

3.1.1 ASFActivation

原型

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

功能描述

激活 SDK。

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，否则返回失败 codes。

3.1.2 ASFInitEngine

原型

```
MRESULT ASFInitEngine(  
    MLong                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,并返回检测结果,需要将所有图像帧的数据都传入接口进行处理; IMAGE 模式:处理单帧的图像数据,并返回检测结果
detectFaceOrientPriority	[in]	检测脸部的角度优先值,推荐仅检测单一角度,效果更优
detectFaceScaleVal	[in]	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] 推荐值为 16
detectFaceMaxNum	[in]	最大需要检测的人脸个数[1-50]
combinedMask	[in]	用户选择需要检测的功能组合,可单个或多个
hEngine	[out]	初始化返回的引擎 handle

返回值

成功返回 MOK, 否则返回失败 codes。

3.1.3 ASFDetectFaces

原型

```
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 handle
width	[in]	图片宽度为 4 的倍数且大于 0
height	[in]	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
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，否则返回失败 codes。

3.1.4 ASFFaceFeatureExtract

原型

```
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 handle
width	[in]	图片宽度为 4 的倍数且大于 0
height	[in]	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
format	[in]	颜色空间格式
imgData	[in]	图片数据
faceInfo	[in]	单张人脸位置和角度信息
feature	[out]	人脸特征

返回值

成功返回 MOK，否则返回失败 codes。

3.1.5 ASFFaceFeatureCompare

原型

```
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 handle
feature1	[in]	待比对的人脸特征
feature2	[in]	待比对的人脸特征
confidenceLevel	[out]	比对结果，置信度数值

返回值

成功返回 MOK，否则返回失败 codes。

3.1.6 ASFProcess

原型

```
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/活体），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）。

参数

hEngine	[in]	引擎 handle
width	[in]	图片宽度为 4 的倍数且大于 0
height	[in]	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

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]	初始化中参数 combinedMask 与 ASF_AGE ASF_GENDER ASF_FACE3DANGLE ASF_LIVENESS 的交集的子集

返回值

成功返回 MOK，否则返回失败 codes。

3.1.7 ASFGetAge

原型

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

功能描述

获取年龄信息。

参数

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

返回值

成功返回 MOK，否则返回失败 codes。

3.1.8 ASFGetGender

原型

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

功能描述

获取性别信息。

参数

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

返回值

成功返回 MOK，否则返回失败 codes。

3.1.9 ASFGetFace3DAngle

原型

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

功能描述

获取 3D 角度信息。

参数

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

返回值

成功返回 MOK，否则返回失败 codes。

3.1.10 ASFGetLivenessScore

原型

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

功能描述

获取活体信息。

参数

hEngine	[in]	引擎 handle
livenessInfo	[out]	检测到的活体信息

返回值

成功返回 MOK，否则返回失败 codes。

3.1.11 ASFGetVersion

原型

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

功能描述

获取版本信息。

参数

hEngine[in] 引擎 handle

返回值

成功返回版本信息，否则返回 MNull。

3.1.12 ASFUninitEngine

原型

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

功能描述

销毁引擎。

参数

hEngine[in] 引擎 handle

返回值

成功返回 MOK，否则返回失败 codes。

3.2 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 AppId
MERR_FSDK_INVALID_SDK_ID	7002	28674	无效的 SDKkey
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_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

3.3 示例代码

```

#include "stdafx.h"
#include <stdio.h>
#include <stdlib.h>
#include "arcsoft_face_sdk.h"
#include "amcomdef.h"
#include "asvloffscreen.h"
#include "merror.h"
#include <direct.h>
#include <iostream>
#include <stdarg.h>
#include <string>
#include <opencv2\opencv.hpp>

using namespace std;
#pragma comment(lib, "libarcsoft_face_engine.lib")

#define APPID "官网下载APPID"
#define SDKKey "官网下载ADKKey"

#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; }

//裁剪图片
void CutIplImage(IplImage* src, IplImage* dst, int x, int y)
{
    CvSize size = cvSize(dst->width, dst->height); //区域大小
    cvSetImageROI(src, cvRect(x, y, size.width, size.height)); //设置源图像ROI
    cvCopy(src, dst); //复制图像
    cvResetImageROI(src); //源图像用完后，清空ROI
}

int main()
{
    //激活接口
    MRESULT res = ASFActivation(APPID, SDKKey);
    if (MOK != res && MERR_ASF_ALREADY_ACTIVATED != res)

```

```

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

//初始化接口
MHandle handle = NULL;
MInt32 mask = ASF_FACE_DETECT | ASF_FACERECOGNITION | ASF_AGE | ASF_GENDER |
ASF_FACE3DANGLE | ASF_LIVENESS;
res = ASFInitEngine(ASF_DETECT_MODE_IMAGE, ASF_OP_0_ONLY, 16, 10, mask, &handle);
if (res != MOK)
    printf("ASFInitEngine fail: %d\n", res);
else
    printf("ASFInitEngine sucess: %d\n", res);

// 人脸检测
IplImage* img0 = cvLoadImage("图片路径");
IplImage* img1 = cvLoadImage("图片路径");

if (img0 && img1)
{
    ASF_MultiFaceInfo detectedFaces1 = { 0 };
    ASF_SingleFaceInfo SingleDetectedFaces1 = { 0 };
    ASF_FaceFeature feature1 = { 0 };
    ASF_FaceFeature copyfeature1 = { 0 };
    IplImage* cutImg0 = cvCreateImage(cvSize(img0->width - img0->width % 4,
img0->height), IPL_DEPTH_8U, img0->nChannels);
    CutIplImage(img0, cutImg0, 0, 0);
    res = ASFDetectFaces(handle, cutImg0->width, cutImg0->height,
ASVL_PAF_RGB24_B8G8R8, (MUInt8*)cutImg0->imageData, &detectedFaces1);
    if (MOK == res)
    {
        SingleDetectedFaces1.faceRect.left = detectedFaces1.faceRect[0].left;
        SingleDetectedFaces1.faceRect.top = detectedFaces1.faceRect[0].top;
        SingleDetectedFaces1.faceRect.right = detectedFaces1.faceRect[0].right;
        SingleDetectedFaces1.faceRect.bottom = detectedFaces1.faceRect[0].bottom;
        SingleDetectedFaces1.faceOrient = detectedFaces1.faceOrient[0];

        res = ASFFaceFeatureExtract(handle, cutImg0->width, cutImg0->height,
ASVL_PAF_RGB24_B8G8R8, (MUInt8*)cutImg0->imageData, &SingleDetectedFaces1, &feature1);
        if (res == MOK)
        {
            //拷贝feature
            copyfeature1.featureSize = feature1.featureSize;
            copyfeature1.feature = (MByte *)malloc(feature1.featureSize);

```

```

        memset(copyfeature1.feature, 0, feature1.featureSize);
        memcpy(copyfeature1.feature, feature1.feature, feature1.featureSize);
    }
    else
        printf("ASFFaceFeatureExtract 1 fail: %d\n", res);
}
else
    printf("ASFDetectFaces 1 fail: %d\n", res);

//第二张人脸提取特征
ASF_MultiFaceInfo detectedFaces2 = { 0 };
ASF_SingleFaceInfo SingleDetectedFaces2 = { 0 };
ASF_FaceFeature feature2 = { 0 };
IplImage* cutImg1 = cvCreateImage(cvSize(img1->width - img1->width % 4,
img1->height), IPL_DEPTH_8U, img1->nChannels);
CutIplImage(img1, cutImg1, 0, 0);
res = ASFDetectFaces(handle, cutImg1->width, cutImg1->height,
ASVL_PAF_RGB24_B8G8R8, (MUInt8*)cutImg1->imageData, &detectedFaces2);
if (MOK == res)
{
    SingleDetectedFaces2.faceRect.left = detectedFaces2.faceRect[0].left;
    SingleDetectedFaces2.faceRect.top = detectedFaces2.faceRect[0].top;
    SingleDetectedFaces2.faceRect.right = detectedFaces2.faceRect[0].right;
    SingleDetectedFaces2.faceRect.bottom = detectedFaces2.faceRect[0].bottom;
    SingleDetectedFaces2.faceOrient = detectedFaces2.faceOrient[0];

    res = ASFFaceFeatureExtract(handle, cutImg1->width, cutImg1->height,
ASVL_PAF_RGB24_B8G8R8, (MUInt8*)cutImg1->imageData, &SingleDetectedFaces2, &feature2);
    if (MOK != res)
        printf("ASFFaceFeatureExtract 2 fail: %d\n", res);
}
else
    printf("ASFDetectFaces 2 fail: %d\n", 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);

```



```
// 人脸信息检测
MInt32 processMask = ASF_AGE | ASF_GENDER | ASF_FACE3DANGLE | ASF_LIVENESS;
res = ASFProcess(handle, cutImg1->width, cutImg1->height, ASVL_PAF_RGB24_B8G8R8,
(MUInt8*)cutImg1->imageData, &detectedFaces2, processMask);
if (res != MOK)
    printf("ASFProcess fail: %d\n", res);
else
    printf("ASFProcess success: %d\n", res);

// 获取年龄
ASF_AgeInfo ageInfo = { 0 };
res = ASFGetAge(handle, &ageInfo);
if (res != MOK)
    printf("ASFGetAge fail: %d\n", res);
else
    printf("ASFGetAge success: %d\n", res);

// 获取性别
ASF_GenderInfo genderInfo = { 0 };
res = ASFGetGender(handle, &genderInfo);
if (res != MOK)
    printf("ASFGetGender fail: %d\n", res);
else
    printf("ASFGetGender success: %d\n", res);

// 获取3D角度
ASF_Face3DAngle angleInfo = { 0 };
res = ASFGetFace3DAngle(handle, &angleInfo);
if (res != MOK)
    printf("ASFGetFace3DAngle fail: %d\n", res);
else
    printf("ASFGetFace3DAngle success: %d\n", res);

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

SafeFree(copyfeature1.feature); //释放内存
cvReleaseImage(&cutImg0);
```

```

        cvReleaseImage(&cutImg1);
    }
    cvReleaseImage(&img0);
    cvReleaseImage(&img1);
    //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 success: %d\n", res);

    getchar();
    return 0;
}

```

4. 常见问题

4.1 FAQ

Q: 初始化引擎时检测方向（detectFaceOrientPriority）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宏有没有在初始化引擎时在 combinedMask 参数中加入。

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。

方案一：ASFDetectFaces 和 ASFFaceFeatureExtract 接口可以在第一次调用时进行数据深拷贝。

方案二：ASFDetectFaces 调用之后进行特征提取（ASFFaceFeatureExtract），如果 ASFDetectFaces 输出结果没用其他用途，则不需要进行深拷贝，但是 ASFFaceFeatureExtract 输出结果需要进行深拷贝用于比对。

Q: 调用 ASFDetectFaces、ASFFaceFeatureExtract 和 ASFProcess 接口返回 90127 错误码，该怎么解决?

A: ArcSoft 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: 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。

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。

4.2 其他帮助

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