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

arm64、armv7

1.2.2 系统要求

iOS 8.x 及以上

1.2.3 开发环境

Xcode 9 及以上

1.2.4 支持的颜色空间格式

NV12, BGR24

常量名	常量值	常量说明
CP_PAF_NV12	2049	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，如图 1 所示。

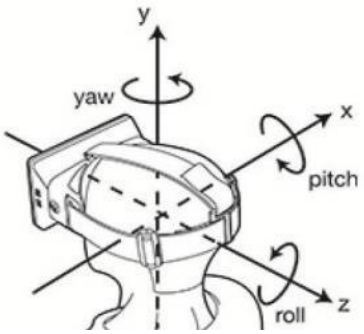


图 1

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。

1.4 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：

- 重刷系统

- 还原系统，还原所有设置

2.接入指南

2.1 引擎获取

2.1.1 注册为开发者

进入 https://www.arcsoft.com.cn/ai/arcface.html 网站，注册账号并登录，进入开发者中心。

2.1.2 创建应用

点击主菜单->应用管理->创建应用，填好应用相关信息后，点击立即创建，创建成功后即如图 2 所示：

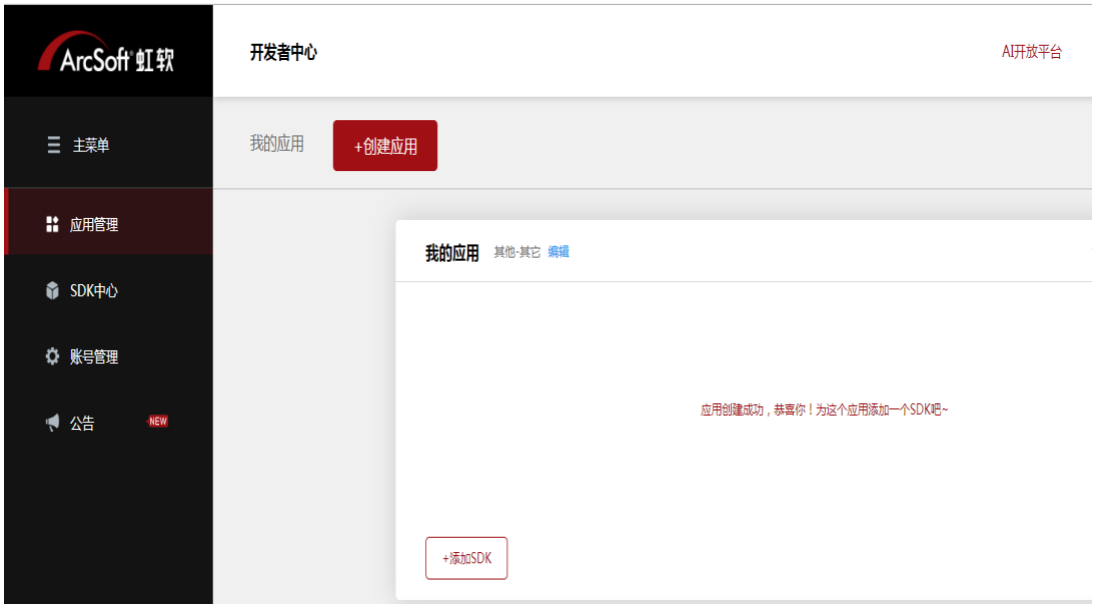


图 2

2.1.3 获取 SDK

点击我的应用->添加 SDK，选择 SDK 平台和 SDK 版本，确认后即可下载 SDK 和查看激活码，如图 3 所示。

图 3

如图 4 所示，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；

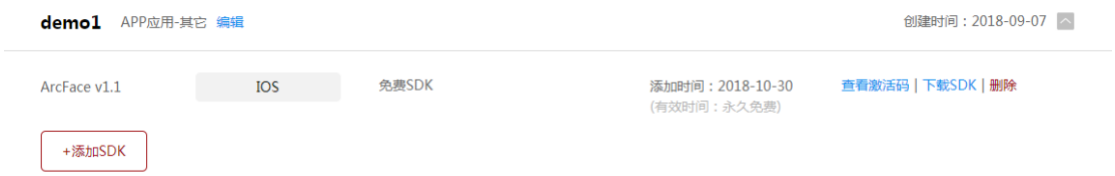


图 4

2.1.4 SDK 包结构

```
|---doc
|    |---appledoc                                离线版appledoc文档
|    |---ARCSOFT_FACE_SDK_DEVELOPER'S_GUIDE.PDF  开发说明文档
|---lib
|    |---ArcSoftFaceEngine.framework             framework文件
|---samplecode
|    |---ArcSoftFaceEngineDemo                   示例代码
|---releasenotes.txt                             说明文件
```

2.2 项目配置

2.2.1 配置 framework 开发包

1、将下载的 SDK 解压，打开 XCode，新建一个 iOS 项目；
2、如图 5 箭头 1 所示，选中左侧目录工程名；

3、如图 5 箭头 2 所示，选中工程所在的 TARGETS，在顶部出现的菜单中选中 Build Phases；

4、如图 5 箭头 3 所示，展开 Link Binary With Libaries 选项，点击“+”号按钮，在弹出的窗口中点击“Add Other”按钮，选择 ArcSoftFaceEngine.framework，添加到工程中；

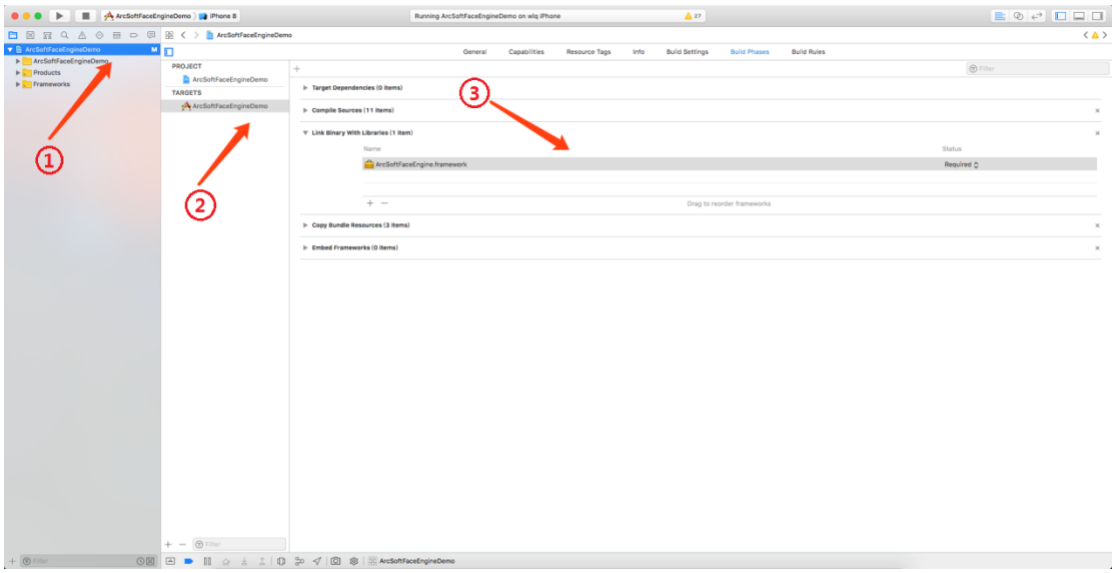


图 5

5、由于 SDK 采用了 Objective-C++实现，需要保证工程中至少有一个.mm 后缀的源文件(可以将任意一个.m 后缀的文件改名为.mm)；或者在工程属性中指定编译方式，即如图 5 箭头 2 所示，选中顶部菜单“Build Setting”，在过滤框中输入“Compile Sources As”，找到 Compile Sources As，并将其设置为"Objective-C++"；

6、需要在在 Xcode 工程中引入系统库：libstdc++.6.0.9.tbd（xcode7 以前为 libstdc++.6.0.9.dylib）：点击 Link Binary With Libaries，点击“+”号添加该系统库即可，如下图 6 中红框所示。

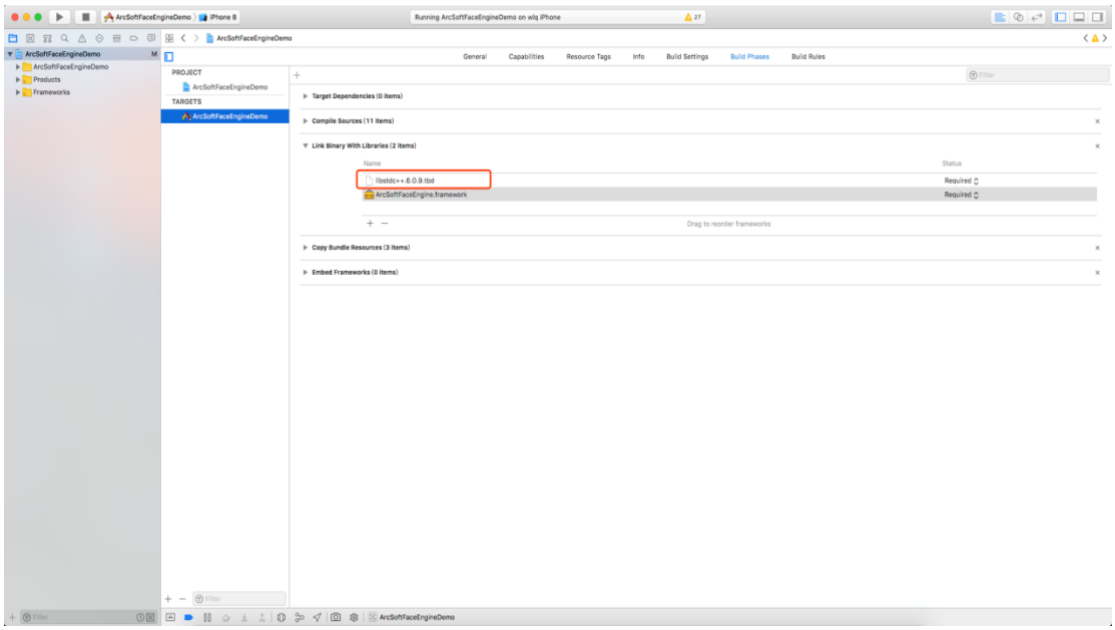


图 6

7、选择项目中的 Info.plist 文件，新增一个属性 App Transport Security Settings，在该属性下添加 Allow Arbitrary Loads 类型 Boolean，值设为 YES，如下图 7 红框所示。

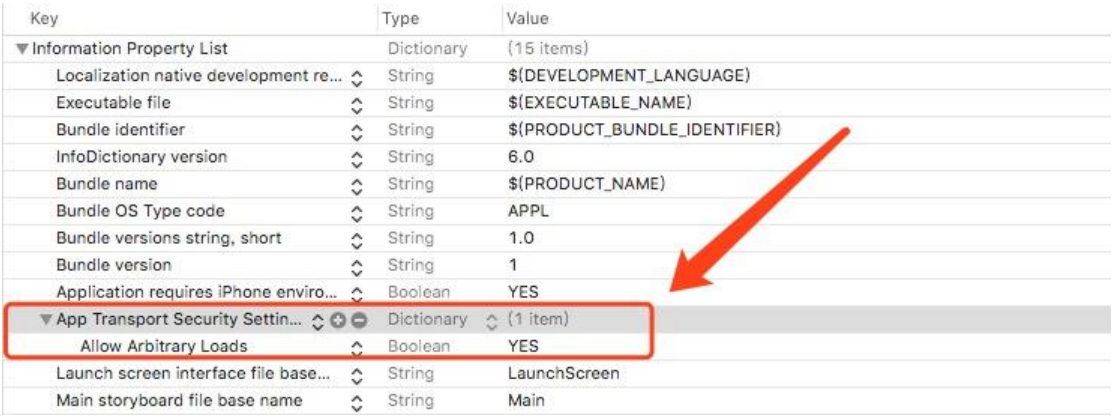


图 7

2.2.2 头文件引入

在使用 SDK 的时候，需要根据情况引入以下头文件：

```
#import <ArcSoftFaceEngine/ArcSoftFaceEngine.h>
#import <ArcSoftFaceEngine/ArcSoftFaceEngineDefine.h>
#import <ArcSoftFaceEngine/amcomdef.h>
#import <ArcSoftFaceEngine/asvloffscreen.h>
#import <ArcSoftFaceEngine/merror.h>
```

2.2.3 头文件简介

2.2.3.1 ArcSoftFaceEngine.h

人脸识别主引擎头文件，主要是对引擎的使用，相关介绍详见 doc 文档。

2.2.3.2 ArcSoftFaceEngineDefine.h

人脸识别主引擎用到的结构体和一些声明头文件，相关介绍详见 doc 文档。

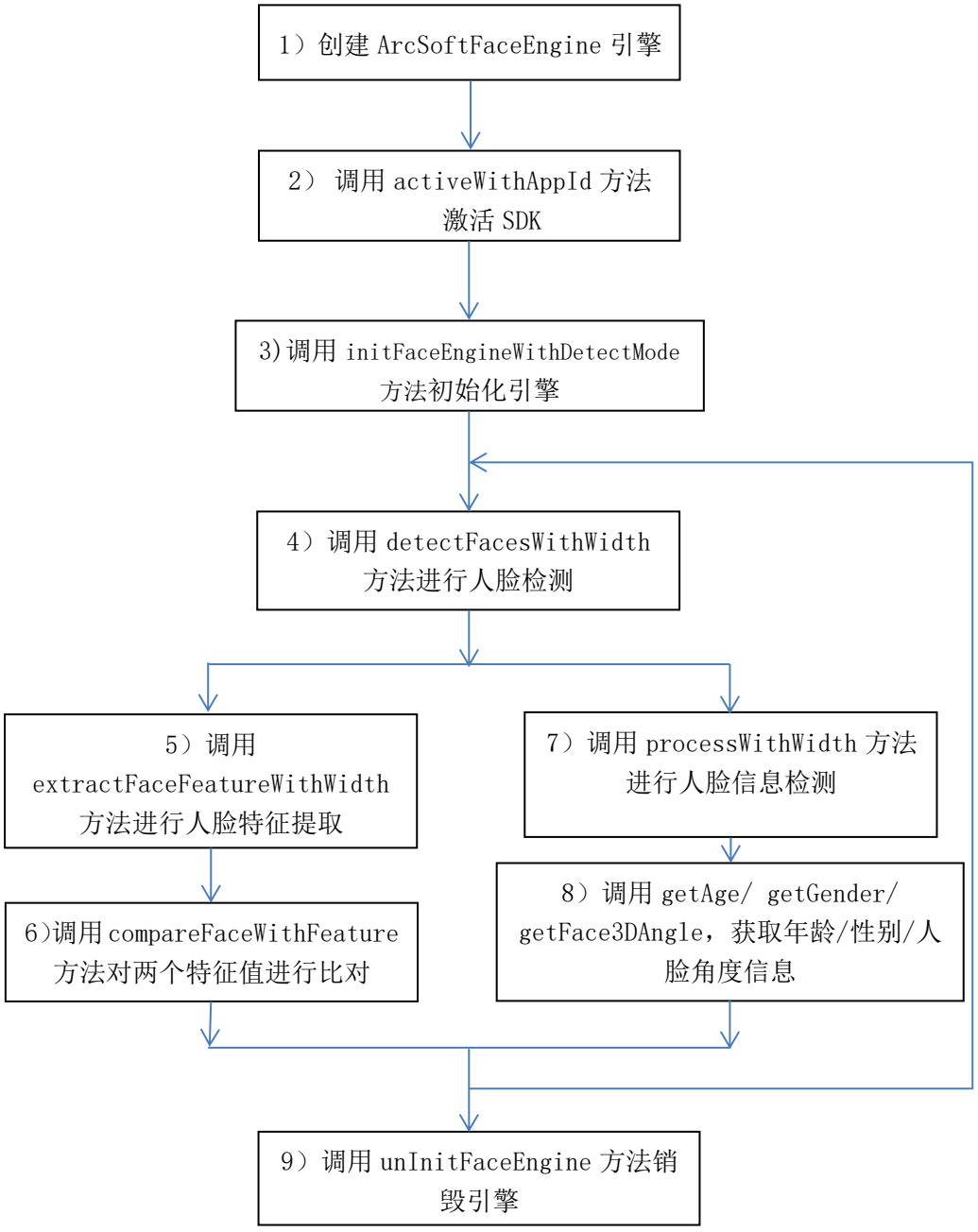
2.2.3.3 amcomdef.h

对 SDK 中使用到的一些数值类型进行二次封装。

2.2.3.4 merror.h

错误码参考。

2.3 调用流程



Step 1：创建 ArcSoftFaceEngine 引擎

Step 2：激活 SDK，调用 activeWithAppId

接口所需 AppId 和 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 3：初始化，调用 initFaceEngineWithDetectMode 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 4: 调用 detectFacesWithWidth 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 5:调用 extractFaceFeatureWithWidth 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 6: 调用 compareFaceWithFeature 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 7: 调用 processWithWidth 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 的交集的子集；

Step 8: 调用 getAge、getGender、getFace3Dangle 接口，年龄、性别、人脸角度信息；

Step 9:调用 unInitEngine 销毁引擎

2.4 通用方法

2.4.1 RGBA 数据转 BGR 数据

```
void RGBA8888ToBGR(unsigned char* pRGBA, int width, int height,int pitch, unsigned char* pBGR) {
    int iSrcXStride = LINE_BYTES(width, 24);
    int iSrcXStride2 = LINE_BYTES(width, 32);
    int i, j;
    for(i = 0; i < height; i++) {
        for(j = 0; j < width; j++) {
            pBGR[i*iSrcXStride+j*3]=pRGBA[i*iSrcXStride2+j*4+2];
            pBGR[i*iSrcXStride+j*3+1]=pRGBA[i*iSrcXStride2+j*4+1];
            pBGR[i*iSrcXStride+j*3+2] = pRGBA[i*iSrcXStride2+j*4];
        }
    }
}
```

2.4.2 BGR 数据转 NV12 数据

```
void BGRToNV12(unsigned char *pBGR, int nW, int nH, unsigned char *nv12Data, int lYStride,
```

```
unsigned char *pYUVUV, int lUVStride)
{
	unsigned int x;
	int y;
	unsigned char *pbSrcX = pBGR;
	unsigned char *pbDstY = nv12Data;
	unsigned char *pbDstUV = pYUVUV;

	int iSrcXStride = LINE_BYTES(nW, 24);
	int iDstYStride = lYStride;
	int iDstUVStride = lUVStride;
	int iSrcXDif;
	int iDstYDif;
	int iDstUVDif;

	iSrcXDif = iSrcXStride - (nW * 3);
	iDstYDif = iDstYStride - nW;
	iDstUVDif = iDstUVStride - nW;

	for (y = nH / 2; y; y--) {
		for (x = nW / 2; x; x--) {
			int r, g, b, y0, y1, y2, y3, cb, cr;

			b = pbSrcX[0];
			g = pbSrcX[1];
			r = pbSrcX[2];
			y0 = yuv_descale(b*yuvYb + g*yuvYg + r*yuvYr);
			cb = yuv_descale((b - y0)*yuvCb) + 128;
			cr = yuv_descale((r - y0)*yuvCr) + 128;

			b = pbSrcX[3];
			g = pbSrcX[4];
			r = pbSrcX[5];
			y1 = yuv_descale(b*yuvYb + g*yuvYg + r*yuvYr);
			cb += yuv_descale((b - y1)*yuvCb) + 128;
			cr += yuv_descale((r - y1)*yuvCr) + 128;

			b = pbSrcX[iSrcXStride    ];
			g = pbSrcX[iSrcXStride+1];
			r = pbSrcX[iSrcXStride+2];
			y2 = yuv_descale(b*yuvYb + g*yuvYg + r*yuvYr);
			cb += yuv_descale((b - y2)*yuvCb) + 128;
			cr += yuv_descale((r - y2)*yuvCr) + 128;
```

```
                b = pbSrcX[iSrcXStride+3];
                g = pbSrcX[iSrcXStride+4];
                r = pbSrcX[iSrcXStride+5];
                y3 = yuv_descale(b*yuvYb + g*yuvYg + r*yuvYr);
                cb += yuv_descale((b - y3)*yuvCb) + 128;
                cr += yuv_descale((r - y3)*yuvCr) + 128;

                pbDstY[0] = ET_CAST_8U(y0);
                pbDstY[1] = ET_CAST_8U(y1);
                pbDstY[iDstYStride    ] = ET_CAST_8U(y2);
                pbDstY[iDstYStride+1] = ET_CAST_8U(y3);
                pbDstUV[0] = ET_CAST_8U(cb>>2);
                pbDstUV[1] = ET_CAST_8U(cr>>2);

                pbSrcX += 6;
                pbDstY += 2;
                pbDstUV += 2;
            }
            pbSrcX += iSrcXDif + iSrcXStride;
            pbDstY += iDstYDif + iDstYStride;
            pbDstUV += iDstUVDif;
        }
}
```

2.4.3 UIImage 转 nv12 数据

```
typedef struct __tag_ ASF_IMAGE_INPUT_DATA
{
    MUInt32      u32PixelArrayFormat;
    MInt32       i32Width;
    MInt32       i32Height;
    MUInt8*      ppu8Plane[4];
    MInt32       pi32Pitch[4];
} IMAGE_INPUT_DATA, * ASF_IMAGE_INPUT_DATA;

+ (unsigned char *) bitmapFromImage: (UIImage *) image {
    CGContextRef context = _fxiang_CreateARGBBitmapContext(image.size);
    if (context == NULL) return NULL;
    CGRect rect = CGRectMake(0.0f, 0.0f, image.size.width, image.size.height);
    CGContextDrawImage(context, rect, image.CGImage);
    unsigned char *data = (unsigned char *)CGBitmapContextGetData (context);
    CGContextRelease(context);
    return data;
}
```

```
void LoadJpg(IMAGE_INPUT_DATA* pSrcImg, NSString *picName) {
    NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES);
    NSString *imagePath = [paths objectAtIndex:0];
    UIImage* imgSrc = [[UIImage alloc]initWithContentsOfFile:[NSString stringWithFormat:@"%@/%@",imagePath,picName]];
    unsigned char* pRGBA = [self bitmapFromImage:imgSrc];
    pSrcImg->i32Width = imgSrc.size.width;
    pSrcImg->i32Height = imgSrc.size.height;
    pSrcImg->u32PixelArrayFormat = ASVL_PAF_NV12;
    pSrcImg->pi32Pitch[0] = pSrcImg->i32Width;
    pSrcImg->pi32Pitch[1] = pSrcImg->i32Width;
    pSrcImg->ppu8Plane[0] = (MUInt8*)malloc(pSrcImg->i32Height*pSrcImg->pi32Pitch[0]*3/2);
    pSrcImg->ppu8Plane[1] = pSrcImg->ppu8Plane[0]+pSrcImg->pi32Pitch[0]*pSrcImg->i32Height;
    unsigned char* pBGR = (unsigned char*)malloc(pSrcImg->i32Height*LINE_BYTES(pSrcImg->i32Width,24));
    RGBA8888ToBGR(pRGBA, pSrcImg->i32Width, pSrcImg->i32Height, pSrcImg->i32Width*4, pBGR);
    BGRToNV12(pBGR, pSrcImg->i32Width, pSrcImg->i32Height, pSrcImg->ppu8Plane[0], pSrcImg->pi32Pitch[0], pSrcImg->ppu8Plane[1], pSrcImg->pi32Pitch[1]);
    SafeArrayFree(pRGBA);
    SafeArrayFree(pBGR);
}
```

2.4.4 从摄像头 CMSampleBufferRef 对象中获取接口需要的数据

```
typedef struct __tag_ASF_CAMERA_DATA
{
    MUInt32     u32PixelArrayFormat;
    MInt32      i32Width;
    MInt32      i32Height;
    MUInt8*     ppu8Plane[4];
    MInt32      pi32Pitch[4];
}ASF_CAMERA_DATA, *ASF_MY_CAMERA_DATA;

+(ASF_MY_CAMERA_DATA)createCameraData:(MInt32)width height:(MInt32) height format:(MUInt32) format {
    ASF_CAMERA_DATA* pCameraData = MNull;
    do {
        pCameraData = (ASF_CAMERA_DATA*)malloc(sizeof(ASF_CAMERA_DATA));
        if(!pCameraData)
```

```
            break;
        memset(pCameraData, 0, sizeof(ASF_CAMERA_DATA));
        pCameraData->u32PixelArrayFormat = format;
        pCameraData->i32Width = width;
        pCameraData->i32Height = height;

        if (ASVL_PAF_NV12 == format) {
            pCameraData->pi32Pitch[0] = pCameraData->i32Width;              //Y
            pCameraData->pi32Pitch[1] = pCameraData->i32Width;              //UV
            pCameraData->ppu8Plane[0]      =      (MUInt8*)malloc(height      *      3/2      *
pCameraData->pi32Pitch[0]) ;       // Y
            pCameraData->ppu8Plane[1]          =          pCameraData->ppu8Plane[0]          +
pCameraData->i32Height * pCameraData->pi32Pitch[0]; // UV
            memset(pCameraData->ppu8Plane[0],         0,         height         *         3/2         *
pCameraData->pi32Pitch[0]);
        } else if (ASVL_PAF_RGB24_B8G8R8 == format) {
            pCameraData->pi32Pitch[0] = pCameraData->i32Width * 3;
            pCameraData->ppu8Plane[0]          =          (MUInt8*)malloc(height          *
pCameraData->pi32Pitch[0]);
        }
    } while(false);
    return pCameraData;
}


+(ASF_CAMERA_INPUT_DATA)getCameraDataFromSampleBuffer:(CMSampleBufferRef)sampleBu
ffer {
    if (NULL == sampleBuffer)
        return NULL;
    CVImageBufferRef cameraFrame = CMSampleBufferGetImageBuffer(sampleBuffer);
    int bufferWidth = (int) CVPixelBufferGetWidth(cameraFrame);
    int bufferHeight = (int) CVPixelBufferGetHeight(cameraFrame);
    OSType pixelType =    CVPixelBufferGetPixelFormatType(cameraFrame);

    CVPixelBufferLockBaseAddress(cameraFrame, 0);
    ASF_CAMERA_DATA* _cameraData;
    if (kCVPixelFormatType_420YpCbCr8BiPlanarVideoRange == pixelType
            || kCVPixelFormatType_420YpCbCr8BiPlanarFullRange == pixelType) {
        _cameraData      =      [Utility      createCameraData:bufferWidth      height:bufferHeight
format:ASVL_PAF_NV12];
        ASF_CAMERA_DATA* pCameraData = _cameraData;
        uint8_t                                *baseAddress0                =                (uint8_t
*)CVPixelBufferGetBaseAddressOfPlane(cameraFrame, 0); // Y
        uint8_t                                *baseAddress1                =                (uint8_t
```

```
*)CVPixelBufferGetBaseAddressOfPlane(cameraFrame, 1); // UV
            size_t      rowBytePlane0 = CVPixelBufferGetBytesPerRowOfPlane(cameraFrame, 0);
            size_t      rowBytePlane1 = CVPixelBufferGetBytesPerRowOfPlane(cameraFrame, 1);

            //Y Data
            if (rowBytePlane0 == pCameraData->pi32Pitch[0]) {
                memcpy(pCameraData->ppu8Plane[0],                                   baseAddress0,
rowBytePlane0*bufferHeight);
            } else {
                for (int i = 0; i < bufferHeight; ++i) {
                    memcpy(pCameraData->ppu8Plane[0] + i * bufferWidth, baseAddress0 + i *
rowBytePlane0, bufferWidth);
                }
            }
            //uv data
            if (rowBytePlane1 == pCameraData->pi32Pitch[1]) {
                memcpy(pCameraData->ppu8Plane[1],      baseAddress1,      rowBytePlane1      *
bufferHeight / 2);
            } else {
                uint8_t    *pPlanUV = pCameraData->ppu8Plane[1];
                for (int i = 0; i < bufferHeight / 2; ++i) {
                    memcpy(pPlanUV  +  i  *  bufferWidth,  baseAddress1+  i  *  rowBytePlane1,
bufferWidth);
                }
            }
        }
        CVPixelBufferUnlockBaseAddress(cameraFrame, 0);
        return _cameraData;
}
```

获取到数据后，可以调用 SDK 中的接口进行人脸检测，代码块如下：

```
ASF_MultiFaceInfo multiFaceInfo = {0};
ASF_CAMERA_DATA* cameraData = [self getCameraDataFromSampleBuffer: sampleBufferRef];
MRESULT mr = [ArcSoftFaceEngine
detectFacesWithWidth:cameraData->i32Width
height: cameraData ->i32Height
data: cameraData ->ppu8Plane[0]
format: cameraData ->u32PixelArrayFormat
faceRes:&multiFaceInfo];
```

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 错误码概览

错误码名	十六进制	十进制	错误码说明
ASF_MOK	0xC8	200	成功
MERR_BASIC_BASE	0x0001	1	通用错误类型
MERR_UNKNOWN	0x0001	1	错误原因不明
MERR_INVALID_PARAM	0x0002	2	无效的参数
MERR_UNSUPPORTED	0x0003	3	引擎不支持
MERR_NO_MEMORY	0x0004	4	内存不足
MERR_BAD_STATE	0x0005	5	状态错误
MERR_USER_CANCEL	0x0006	6	用户取消相关操作
MERR_EXPIRED	0x0007	7	操作时间过期
MERR_USER_PAUSE	0x0008	8	用户暂停操作
MERR_BUFFER_OVERFLOW	0x0009	9	缓冲上溢
MERR_BUFFER_UNDERFLOW	0x000A	10	缓冲下溢
MERR_NO_DISKSPACE	0x000B	11	存贮空间不足
MERR_COMPONENT_NOT_EXIST	0x000C	12	组件不存在
MERR_GLOBAL_DATA_NOT_EXIST	0x000D	13	全局数据不存在
MERR_ASF_SDK_BASE	0x7000	28672	FreeSDK 通用错误类型
MERR_ ASF_SDK_INVALID_APP_ID	0x7001	28673	无效的 App Id
MERR_ ASF_SDK_INVALID_SDK_ID	0x7002	28674	无效的 SDK key

MERR_ ASF_SDK_INVALID_ID_PAIR	0x7003	28675	AppId 和 SDKKey 不匹配
MERR_ ASF_SDK_MISMATCH_ID_AND_SDK	0x7004	28676	SDKKey 和使用的 SDK 不匹配
MERR_ ASF_SDK_SYSTEM_VERSION_UNSUPPORTED	0x7005	28677	系统版本不被当前 SDK 所支持
MERR_ ASF_SDK_LICENCE_EXPIRED	0x7006	28678	SDK 有效期过期
MERR_ASF_SDK_FR_ERROR_BASE	0x12000	73728	FaceRecognition 错误类型
MERR_ASF_SDK_FR_INVALID_MEMORY_INFO	0x12001	73729	无效的输入内存
MERR_ASF_SDK_FR_INVALID_IMAGE_INFO	0x12002	73730	无效的输入图像参数
MERR_ASF_SDK_FR_INVALID_FACE_INFO	0x12003	73731	无效的脸部信息
MERR_ASF_SDK_FR_NO_GPU_AVAILABLE	0x12004	73732	当前设备无 GPU 可用
MERR_ASF_SDK_FR_MISMATCHED_FEATURE_LEVEL	0x12005	73733	待比较的两个人脸特征的版本不一致
MERR_ ASF_SDK_FACEFEATURE_ERROR_BASE	0x14000	81920	人脸特征检测错误类型
MERR_ ASF_SDK_FACEFEATURE_UNKNOWN	0x14001	81921	人脸特征检测错误未知
MERR_ ASF_SDK_FACEFEATURE_MEMORY	0x14002	81922	人脸特征检测内存错误
MERR_ ASF_SDK_FACEFEATURE_INVALID_FORMAT	0x14003	81923	人脸特征检测格式错误
MERR_ ASF_SDK_FACEFEATURE_INVALID_PARAM	0x14004	81924	人脸特征检测参数错误
MERR_ ASF_SDK_FACEFEATURE_LOW_CONFIDENCE_LEVEL	0x14005	81925	人脸特征检测结果置信度低
MERR_ASF_EX_BASE	0x15000	86016	ArcFace 扩展错误类型
MERR_ASF_EX_BASE_FEATURE_UNSUPPORTED_ON_INIT	0x15001	86017	Engine 不支持的检测属性

MERR_ASF_EX_BASE_FEATURE_UNINITED	0x15002	86018	需要检测是属性未初始化
MERR_ASF_EX_BASE_FEATURE_UNPROCESSED	0x15003	86019	待获取的属性未在 process 中处理过
MERR_ASF_EX_BASE_FEATURE_UNSUPPORTED_ON_PROCESS	0x15004	86020	PROCESS 不支持的检测属性
MERR_ASF_EX_BASE_INVALID_IMAGE_INFO	0x15005	86021	无效的输入图像
MERR_ASF_EX_BASE_INVALID_FACE_INFO	0x15006	86022	无效的脸部信息
MERR_ASF_BASE	0x16000	90112	人脸比对基础错误类型
MERR_ASF_BASE_ACTIVATION_FAIL	0x16001	90113	人脸比对 SDK 激活失败
MERR_ASF_BASE_ALREADY_ACTIVATED	0x16002	90114	人脸比对 SDK 已激活
MERR_ASF_BASE_NOT_ACTIVATED	0x16003	90115	人脸比对 SDK 未激活
MERR_ASF_BASE_SCALE_NOT_SUPPORT	0x16004	90116	detectFaceScaleVal 不支持
MERR_ASF_BASE_VERION_MISMATCH	0x16005	90117	SDK 版本不匹配
MERR_ASF_BASE_DEVICE_MISMATCH	0x16006	90118	设备不匹配
MERR_ASF_BASE_UNIQUE_IDENTIFIER_MISMATCH	0x16007	90119	唯一标识不匹配
MERR_ASF_BASE_PARAM_NULL	0x16008	90120	参数为空
MERR_ASF_BASE_SDK_EXPIRED	0x16009	90121	SDK 已过期
MERR_ASF_BASE_VERSION_NOT_SUPPORT	0x1600A	90122	版本不支持
MERR_ASF_BASE_SIGN_ERROR	0x1600B	90123	签名错误
MERR_ASF_BASE_DATABASE_ERROR	0x1600C	90124	数据库插入错误
MERR_ASF_BASE_UNIQUE_CHECKOUT_FAIL	0x1600D	90125	唯一标识符校验失败
MERR_ASF_BASE_COLOR_SPACE_NOT_SUPPORT	0x1600E	90126	输入的颜色空间不支持
MERR_ASF_IMAGE_WIDTH_HEIGHT_NOT_SUPPORT	0x1600F	90127	图片宽高不支持
MERR_ASF_NETWORK_BASE	0x17000	94208	网络错误类型

MERR_ASF_NETWORK_BASE_SERVER_EXCEPTION	0x17001	94209	服务器异常
MERR_ASF_NETWORK_BASE_CONNECT_TIMEOUT	0x17002	94210	网络请求超时
MERR_ASF_NETWORK_BASE_NONSUPPORTED_URL	0x17003	94211	不支持的 URL
MERR_ASF_NETWORK_BASE_COULDNT_RESOLVE_HOST	0x17004	94212	未能找到指定的服务器
MERR_ASF_NETWORK_BASE_CONNECT_FAILURE	0x17005	94213	服务器连接失败
MERR_ASF_NETWORK_BASE_CONNECT_LOSS	0x17006	94214	连接丢失
MERR_ASF_NETWORK_BASE_NOT_CONNECTED	0x17007	94215	连接中断
MERR_ASF_NETWORK_BASE_OPERATION_NOT_COMPLETED	0x17008	94216	操作无法完成
MERR_ASF_NETWORK_BASE_UNKNOWN_ERROR	0x17009	94217	未知错误

3.2 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: 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。

3.3 其他帮助

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