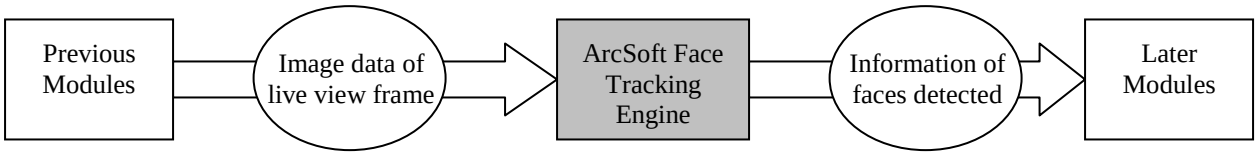
ArcSoft Face Tracking

开发指导文档

概述

虹软人脸跟踪引擎工作流程图:



1.1. 运行环境

- iOS Arm v7, Arm 64

1.2. 系统要求

- 支持 iOS 8.x 或以上

1.3. 依赖库

- 虹软平台库

注: 请把虹软平台库的头文件 (在 SDK 包的“platform”目录下)放入您的开发工程里面。

结构与变量

2.1. 基本类型

```
typedef  MInt32            AFT_FSDK_OrientPriority;
typedef  MInt32            AFT_FSDK_OrientCode;
```

所有基本类型在平台库中有定义。 定义规则是在 ANSIC 中的基本类型前加上字母“M”同时将类型的第一个字母改成大写。例如“long” 被定义成“MLong”。

2.2. 数据结构与枚举

2.2.1. AFT_FSDK_FACERES

描述

检测到的脸部信息。

定义

```
typedef struct{
       MInt32 nFace;
       AFT_OrientCode lfaceOrient;
       MRECT  *rcFace;
} AFT_FSDK_FACERES, *LPAFT_FSDK_FACERES;
```

成员描述

rcFace	人脸矩形框信息
nFace	人脸个数
lfaceOrient	人脸角度信息

2.2.2. AFT_FSDK_Version

描述

SDK 版本信息。

定义

```
typedef struct{
          MInt32 lCodebase;
          MInt32 lMajor;
          MInt32 lMinor;
```

```
        MInt32 lBuild;
        MPChar Version;
        MPChar BuildDate;
        MPChar CopyRight;
} AFT_FSDK_Version;
```

成员描述

lCodebase	代码库版本号
lMajor	主版本号
lMinor	次版本号
lBuild	编译版本号，递增
Version	字符串形式的版本号
BuildDate	编译时间
CopyRight	版权

2.2.3. AFT_FSDK_OrientPriority

描述

定义脸部检测角度的优先级 。

定义

```
enum _AFT_FSDK_OrientPriority {
       AFT_FSDK_OPF_0_ONLY          = 0x1,
       AFT_FSDK_OPF_90_ONLY         = 0x2,
       AFT_FSDK_OPF_270_ONLY        = 0x3,
       AFT_FSDK_OPF_180_ONLY        = 0x4,
       AFT_FSDK_OPF_0_HIGHER_EXT    = 0x5,
};
```

成员描述

AFT_FSDK_OPF_0_ONLY	检测 0 度方向
AFT_FSDK_OPF_90_ONLY	检测 90 度方向
AFT_FSDK_OPF_270_ONLY	检测 270 度方向
AFT_FSDK_OPF_180_ONLY	检测 180 度方向
AFT_FSDK_OPF_0_HIGHER_EXT	检测 0， 90， 180， 270 四个方向, 其中 0 度更优先

2.2.4. AFT_FSDK_OrientCode

描述

定义检测结果中的人脸角度。

定义

```
enum _AFT_FSDK_OrientCode {
       AFT_FSDK_FOC_0                = 0x1,
       AFT_FSDK_FOC_90               = 0x2,
       AFT_FSDK_FOC_270              = 0x3,
       AFT_FSDK_FOC_180               = 0x4
};
```

成员描述

AFT_FSDK_FOC_0	0 度
AFT_FSDK_FOC_90	90 度
AFT_FSDK_FOC_270	270 度
AFT_FSDK_FOC_180	180 度

2.2.5. 支持的颜色格式

描述

颜色格式及其对齐规则

定义

ASVL_PAF_NV12	8-bit Y 层，之后是 8-bit 的 2x2 采样的 U 层和 V 层
ASVL_PAF_RGB24_B8G8R8	每个像素 8-bit B，8-bit R，8-bit R

2.2.6. 错误码

具体的错误码定义可以参考平台库中 merror.h 文件

API 说明

3.1. AFT_FSDK_InitialFaceEngine

原型

```
MRESULT AFT_FSDK_InitialFaceEngine(
       MPChar               AppId,
       MPChar               SDKKey,
       MByte*               pMem,
       MLong                lMemSize,
       MHandle              *phEngine,
       AFT_FSDK_OrientPriority iOrientPriority,
       MInt32             nScale,
       MInt32             nMaxFaceNum
);
```

描述

初始化人脸跟踪引擎

参数

AppId	[in]	用户申请 SDK 时获取的 App Id
SDKKey	[in]	用户申请 SDK 时获取的 SDK Key
pMem	[in]	分配给引擎使用的内存地址
lMemSize	[in]	分配给引擎使用的内存大小
phEngine	[out]	引擎句柄
iOrientPriority	[in]	期望的脸部检测角度的优先级
nScale	[in]	用于数值表示的最小人脸尺寸 有效值范围[2,16] 推荐值 16
nMaxFaceNum	[in]	用户期望引擎最多能检测出的人脸数 有效值范围[1,20]

返回值

成功返回 MOK，否则返回失败 code。失败 codes 如下所列:

MERR_INVALID_PARAM	参数输入非法
MERR_NO_MEMORY	内存不足

3.2. AFT_FSDK_FaceFeatureDetect

原型

```
MRESULT AFT_FSDK_FaceFeatureDetect(
       MHandle              hEngine,
       LPASVLOFFSCREEN      pImgData,
       LPAFT_FSDK_FACERES   *pFaceRes
);
```

描述

根据输入的图像检测人脸，一般用于视频检测，多帧方式。

参数

hEngine	[in]	引擎句柄
pImgData	[in]	带检测图像信息
pFaceRes	[out]	人脸检测结果

说明

在一个 init/unit 过程中， 只支持相同分辨率的图像数据

返回值

成功返回 MOK，否则返回失败 code。失败 codes 如下所列:

MERR_INVALID_PARAM	参数输入非法
MERR_NO_MEMORY	内存不足
MERR_BAD_STATE	状态不正确

3.3. AFT_FSDK_UninitialFaceEngine

原型

```
MRESULT AFT_FSDK_UninitialFaceEngine(
       MHandle         hEngine
);
```

描述

销毁引擎，释放相应资源。

参数

hEngine	[in]	引擎句柄

返回值

成功返回 MOK，否则返回失败 code。失败 codes 如下所列:

MERR_INVALID_PARAM	参数输入非法

3.4. AFT_FSDK_GetVersion

原型

```
const AFT_FSDK_Version * AFT_FSDK_GetVersion (
      MHandle         hEngine
);
```

描述

获取 SDK 版本信息。

参数

hEngine	[in]	引擎句柄

示例代码

```
#include "ammem.h"
#include "merror.h"
#import <arcsoft_fsdk_face_tracking/arcsoft_fsdk_face_tracking.h>

#include <stdlib.h>


#define ARC_APP_ID              ""
#define ARC_FT_SDK_KEY          ""
#define ARC_FT_MAX_FACE_NUM     5
#define ARC_FT_MEM_SIZE         1024*1024*5

MRESULT doFaceTracking()
{

    MVoid* pMemBuffer = MMemAlloc(MNull, ARC_FT_MEM_SIZE);
    MHandle hEngine = MNull;

     //初始化人脸检测跟踪引擎
    MRESULT mr = AFT_FSDK_InitialFaceEngine((MPChar)ARC_APP_ID,
(MPChar)ARC_FT_SDK_KEY, (MByte*)pMemBuffer, ARC_FT_MEM_SIZE, &hEngine,
AFT_FSDK_OPF_0_HIGHER_EXT, 16, ARC_FT_MAX_FACE_NUM);
    if (MOK != mr) {
        //错误码检查
    }

    // 连续视频图像数据
    do {
        ASVLOFFSCREEN offScreenIn = {0};

       //指定图像数据格式
        offScreenIn.u32PixelArrayFormat = ASVL_PAF_NV12;
        offScreenIn.i32Width = 1280;
        offScreenIn.i32Height = 720;
        offScreenIn.pi32Pitch[0] = offScreenIn.i32Width;
        offScreenIn.pi32Pitch[1] = offScreenIn.i32Width;
        offScreenIn.ppu8Plane[0] = MNull;
        offScreenIn.ppu8Plane[1] = MNull;

        LPAFT_FSDK_FACERES pFaceRes = MNull;

       //检测人脸，并把结果输出到pFaceRes
        mr = AFT_FSDK_FaceFeatureDetect(hEngine, &offScreenIn, &pFaceRes);
    } while (MFalse);
```

```
    //对人脸检测跟踪引擎做销毁
    mr = AFT_FSDK_UninitialFaceEngine(&hEngine);

    if(pMemBuffer != MNull)
    {
        MMemFree(MNull,pMemBuffer);
        pMemBuffer = MNull;
    }

    return mr;
}
```