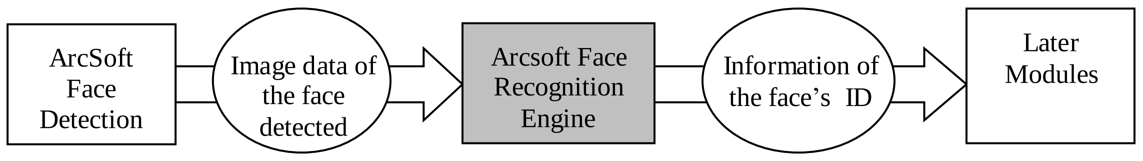
ArcSoft Face Recognition

开发指导文档

ArcSoft Corporation
46601 Fremont Blvd.
Fremont, CA 94538
http://www.arcsoft.com

1. 概述

虹软人脸识别引擎工作流程图：


ArcSoft Face Detection
Image data of the face detected
Arcsoft Face Recognition Engine
Information of the face’s ID
Later Modules

1.1. 运行环境

- Linux x64

1.2. 系统要求

- 库依赖 GLIBC 2.19 及以上
- 编译器 GCC 4.8.2 及以上

1.3. 依赖库

- libsqlite3
- libcurl

2. 结构与常量

2.1. 基本类型

```
typedef MInt32 AFR_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. AFR_FSDK_FACEINPUT

功能描述

脸部信息

定义

```
typedef struct{
       MRECT                rcFace;
       AFR_FSDK_ORIENTCODE lOrient;
} AFR_FSDK_FACEINPUT, *LPAFR_FSDK_FACEINPUT;
```

成员变量

rcFace	脸部矩形框信息
lOrient	脸部旋转角度

2.2.2. AFR_FSDK_FACEMODEL

功能描述

脸部特征信息

定义

```
typedef struct{
       MByte        *pbFeature;
       MInt32       lFeatureSize;
} AFR_FSDK_FACEMODEL, *LPAFR_FSDK_FACEMODEL;
```

成员变量

pbFeature	提取到的脸部特征
lFeatureSize	特征信息长度

2.2.3. AFR_FSDK_VERSION

功能描述

SDK 版本信息.

定义

```
typedef struct{
       MInt32              lCodebase;
       MInt32              lMajor;
       MInt32              lMinor;
       MInt32              lBuild;
       MInt32              lFeatureLevel;
       MPChar              Version;
       MPChar              BuildDate;
       MPChar              CopyRight;
} AFR_FSDK_VERSION, *LPAFR_FSDK_VERSION;
```

成员变量

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
lFeatureLevel	特征库版本号
Version	字符串形式的版本号
BuildDate	编译时间
CopyRight	Copyright

2.2.4. AFR_FSDK_ORIENTCODE

功能描述

基于逆时针的脸部方向枚举值

定义

```
enum _AFR_FSDK_ORIENTCODE{
       AFR_FSDK_FOC_0            = 0x1,
       AFR_FSDK_FOC_90           = 0x2,
       AFR_FSDK_FOC_270          = 0x3,
       AFR_FSDK_FOC_180          = 0x4,
       AFR_FSDK_FOC_30           = 0x5,
       AFR_FSDK_FOC_60           = 0x6,
        AFR_FSDK_FOC_120            = 0x7,
        AFR_FSDK_FOC_150            = 0x8,
        AFR_FSDK_FOC_210            = 0x9,
        AFR_FSDK_FOC_240            = 0xa,
        AFR_FSDK_FOC_300            = 0xb,
        AFR_FSDK_FOC_330            = 0xc
};
```

成员变量

AFR_FSDK_FOC_0	0 度
AFR_FSDK_FOC_90	90 度
AFR_FSDK_FOC_270	270 度
AFR_FSDK_FOC_180	180 度
AFR_FSDK_FOC_30	30 度
AFR_FSDK_FOC_60	60 度
AFR_FSDK_FOC_120	120 度
AFR_FSDK_FOC_150	150 度
AFR_FSDK_FOC_210	210 度
AFR_FSDK_FOC_240	240 度
AFR_FSDK_FOC_300	300 度
AFR_FSDK_FOC_330	330 度

2.2.5. 支持的颜色格式

定义	说明
ASVL_PAF_I420	8-bit Y 通道，8-bit 2x2 采样 U 通道，8-bit 2x2 采样 V 通道
ASVL_PAF_NV12	8-bit Y 通道，8-bit 2x2 采样 U 与 V 分量交织通道
ASVL_PAF_NV21	8-bit Y 通道，8-bit 2x2 采样 V 与 U 分量交织通道
ASVL_PAF_YUYV	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_RGB24_B8G8R8	RGB 分量交织，按 B, G, R, B 字节序排布

3. API Reference

3.1. AFR_FSDK_InitialEngine

原型

```
MRESULT AFR_FSDK_InitialEngine(
       MPChar       AppId,
       MPChar       SDKKey,
       Mbyte        *pMem,
       MInt32       lMemSize,
       MHandle      *phEngine
);
```

功能描述

初始化引擎

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 handle

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. AFR_FSDK_ExtractFRFeature

原型

```
MRESULT AFR_FSDK_ExtractFRFeature (
       MHandle                    hEngine,
       LPASVLOFFSCREEN            pInputImage,
       LPAFR_FSDK_FACEINPUT       pFaceRes,
       LPAFR_FSDK_FACEMODEL       pFaceModels
);
```

功能描述

获取脸部特征

参数

hEngine	[in]	引擎 handle
pInputImage	[in]	输入的图像数据
pFaceRes	[in]	已检测到到的脸部信息
pFaceModels	[out]	提取到的脸部特征信息

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

3.3. AFR_FSDK_FacePairMatching

原型

```
MRESULT AFR_FSDK_FacePairMatching(
      MHandle                    hEngine,
      AFR_FSDK_FACEMODEL         *reffeature,
      AFR_FSDK_FACEMODEL         *probefeature,
      MFloat                     *pfSimilScore
);
```

功能描述

脸部特征比较.

参数

hEngine	[in]	引擎 handle
reffeature	[in]	已有脸部特征信息
probefeature	[in]	被比较的脸部特征信息
pfSimilScore	[out]	相似程度数值

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

3.4. AFR_FSDK_UninitialEngine

原型

```
MRESULT AFR_FSDK_UninitialEngine(
       MHandle          hEngine
);
```

功能描述

销毁引擎，释放相应资源

参数

`hEngine`	`[in]`	引擎 `handle`

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

3.5. AFR_FSDK_GetVersion

原型

```
const AFR_FSDK_VERSION *  AFR_FSDK_GetVersion(MHandle          hEngine);
```

功能描述

获取引擎版本信息

参数

`hEngine`	`[in]`	引擎 `handle`

功能描述

获取引擎版本信息

参数

None

4. 示例代码

注意,使用时请替换申请的 **APPID** 和 **SDKKEY**，并设置好文件路径和图像尺寸

```
#include <stdlib.h>
#include <stdio.h>
#include <stdint.h>
#include <string.h>
#include <errno.h>
#include <assert.h>


#include "arcsoft_fsdk_face_recognition.h"
#include "merror.h"

//#define APPID     "your appid"
//#define SDKKEY    "your sdkkey"

#define INPUT1_IMAGE_FORMAT  ASVL_PAF_I420
//#define INPUT1_IMAGE_PATH    "your_input1_image.yuv"
#define INPUT1_IMAGE_WIDTH   (640)
#define INPUT1_IMAGE_HEIGHT  (480)

#define INPUT2_IMAGE_FORMAT  ASVL_PAF_I420
//#define INPUT2_IMAGE_PATH    "your_input2_image.yuv"
#define INPUT2_IMAGE_WIDTH   (640)
#define INPUT2_IMAGE_HEIGHT  (880)

#define WORKBUF_SIZE        (40*1024*1024)

int fu_ReadFile(const char* path, uint8_t **raw_data, size_t* pSize) {
    int res = 0;
    FILE *fp = 0;
    uint8_t *data_file = 0;
    size_t size = 0;

    fp = fopen(path, "rb");
    if (fp == nullptr) {
        res = -1;
        goto exit;
    }

    fseek(fp, 0, SEEK_END);
    size = ftell(fp);
    fseek(fp, 0, SEEK_SET);

    data_file = (uint8_t *)malloc(sizeof(uint8_t)* size);
    if (data_file == nullptr) {
        res = -2;
        goto exit;
    }
    if (size != fread(data_file, sizeof(uint8_t), size, fp)) {
        res = -3;
        goto exit;
    }

    *raw_data = data_file;
    data_file = nullptr;
exit:
    if (fp != nullptr) {
        fclose(fp);
    }

    if (data_file != nullptr) {
        free(data_file);
    }

    if (nullptr != pSize) {
        *pSize = size;
    }

    return res;
}


int main(int argc, char* argv[]) {

    MByte *pWorkMem = (MByte *)malloc(WORKBUF_SIZE);
    if(pWorkMem == nullptr){
        fprintf(stderr, "fail to malloc workbuf\r\n");
        exit(0);
    }

    MHandle hEngine = nullptr;

    int ret = AFR_FSDK_InitialEngine(APPID, SDKKEY, pWorkMem, WORKBUF_SIZE,
&hEngine);
    if (ret != 0) {
        fprintf(stderr, "fail to AFR_FSDK_InitialEngine(): 0x%x\r\n", ret);
        free(pWorkMem);
        exit(0);
    }

   const AFR_FSDK_Version*pVersionInfo = AFR_FSDK_GetVersion(hEngine);
    printf("%d %d %d %d\r\n", pVersionInfo->lCodebase, pVersionInfo->lMajor,
                               pVersionInfo->lMinor, pVersionInfo->lBuild);
    printf("%s\r\n", pVersionInfo->Version);
    printf("%s\r\n", pVersionInfo->BuildDate);
    printf("%s\r\n", pVersionInfo->CopyRight);

    ASVLOFFSCREEN inputImg1 = { 0 };
    inputImg1.u32PixelArrayFormat = INPUT1_IMAGE_FORMAT;
    inputImg1.i32Width = INPUT1_IMAGE_WIDTH;
    inputImg1.i32Height = INPUT1_IMAGE_HEIGHT;
    inputImg1.ppu8Plane[0] = nullptr;
    fu_ReadFile(INPUT1_IMAGE_PATH, (uint8_t**)&inputImg1.ppu8Plane[0],
nullptr);
     if (!inputImg1.ppu8Plane[0]) {
        fprintf(stderr, "fail to fu_ReadFile(%s): %s\r\n", INPUT1_IMAGE_PATH,
strerror(errno));
        AFR_FSDK_UninitialEngine(hEngine);
        free(pWorkMem);
        exit(0);
    }
    inputImg1.pi32Pitch[0] = inputImg1.i32Width;
    inputImg1.pi32Pitch[1] = inputImg1.i32Width/2;
    inputImg1.pi32Pitch[2] = inputImg1.i32Width/2;
    inputImg1.ppu8Plane[1] = inputImg1.ppu8Plane[0] + inputImg1.pi32Pitch[0] *
inputImg1.i32Height;
    inputImg1.ppu8Plane[2] = inputImg1.ppu8Plane[1] + inputImg1.pi32Pitch[1] *
inputImg1.i32Height/2;


    ASVLOFFSCREEN inputImg2 = { 0 };
    inputImg2.u32PixelArrayFormat = INPUT2_IMAGE_FORMAT;
    inputImg2.i32Width = INPUT2_IMAGE_WIDTH;
    inputImg2.i32Height = INPUT2_IMAGE_HEIGHT;
    inputImg2.ppu8Plane[0] = nullptr;
    fu_ReadFile(INPUT2_IMAGE_PATH, (uint8_t**)&inputImg2.ppu8Plane[0],
nullptr);
     if (!inputImg2.ppu8Plane[0]) {
        fprintf(stderr, "fail to fu_ReadFile(%s): %s\r\n", INPUT2_IMAGE_PATH,
strerror(errno));
         free(inputImg1.ppu8Plane[0]);
         AFR_FSDK_UninitialEngine(hEngine);
        free(pWorkMem);
        exit(0);
    }
    inputImg2.pi32Pitch[0] = inputImg2.i32Width;
    inputImg2.pi32Pitch[1] = inputImg2.i32Width/2;
    inputImg2.pi32Pitch[2] = inputImg2.i32Width/2;
    inputImg2.ppu8Plane[1] = inputImg2.ppu8Plane[0] + inputImg2.pi32Pitch[0] *
inputImg2.i32Height;
    inputImg2.ppu8Plane[2] = inputImg2.ppu8Plane[1] + inputImg2.pi32Pitch[1] *
inputImg2.i32Height/2;

    AFR_FSDK_FACEMODEL faceModels1 = { 0 };
    {
        AFR_FSDK_FACEINPUT faceResult;
        faceResult.lOrient = AFR_FSDK_FOC_0;
        faceResult.rcFace.left = 282;
        faceResult.rcFace.top = 58;
        faceResult.rcFace.right = 422;
        faceResult.rcFace.bottom = 198;
        AFR_FSDK_FACEMODEL LocalFaceModels = { 0 };
        ret = AFR_FSDK_ExtractFRFeature(hEngine, &inputImg1, &faceResult,
&LocalFaceModels);
        if(ret != 0){
            fprintf(stderr, "fail to AFR_FSDK_ExtractFRFeature in Image
A\r\n");
```

```
            free(inputImg2.ppu8Plane[0]);
            free(inputImg1.ppu8Plane[0]);
            AFR_FSDK_UninitialEngine(hEngine);
            free(pWorkMem);
            exit(0);
        }

        faceModels1.lFeatureSize = LocalFaceModels.lFeatureSize;
        faceModels1.pbFeature = (MByte*)malloc(faceModels1.lFeatureSize);
        memcpy(faceModels1.pbFeature, LocalFaceModels.pbFeature,
faceModels1.lFeatureSize);
        free(inputImg1.ppu8Plane[0]);
    }

    AFR_FSDK_FACEMODEL faceModels2 = { 0 };
    {
        AFR_FSDK_FACEINPUT faceResult;
        faceResult.lOrient = AFR_FSDK_FOC_0;
        faceResult.rcFace.left = 176;
        faceResult.rcFace.top = 57;
        faceResult.rcFace.right = 269;
        faceResult.rcFace.bottom = 151;
        AFR_FSDK_FACEMODEL LocalFaceModels = { 0 };
        ret = AFR_FSDK_ExtractFRFeature(hEngine, &inputImg2, &faceResult,
&LocalFaceModels);
        if(ret != 0){
            fprintf(stderr, "fail to AFR_FSDK_ExtractFRFeature in Image
B\r\n");
            free(inputImg2.ppu8Plane[0]);
            AFR_FSDK_UninitialEngine(hEngine);
            free(pWorkMem);
            exit(0);
        }

        faceModels2.lFeatureSize = LocalFaceModels.lFeatureSize;
        faceModels2.pbFeature = (MByte*)malloc(faceModels2.lFeatureSize);
        memcpy(faceModels2.pbFeature, LocalFaceModels.pbFeature,
faceModels2.lFeatureSize);
        free(inputImg2.ppu8Plane[0]);
    }

    MFloat  fSimilScore = 0.0f;
    ret = AFR_FSDK_FacePairMatching(hEngine, &faceModels1, &faceModels2,
&fSimilScore);
    printf("fSimilScore ==  %f\r\n", fSimilScore);

    free(faceModels1.pbFeature);
    free(faceModels2.pbFeature);
    AFR_FSDK_UninitialEngine(hEngine);
    free(pWorkMem);

    return 0;
}
```

5. 其他说明

此版本为免费开放的标准版本(为保证最优体验，建议注册人脸数小于 1000)，若有定制升级需求，请联系我们。