ArcSoft, Inc.

ArcSoft Face Recognition

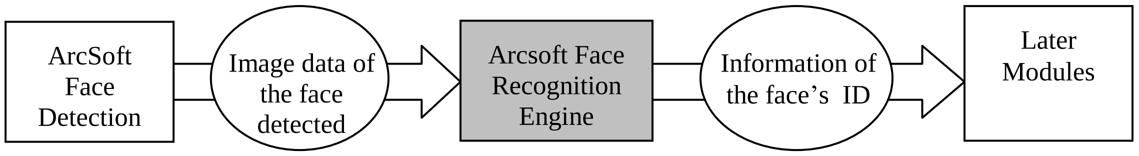
开发指导文档

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

Trademark or Service Mark Information

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

- Windows

1.2. 系统要求

- 32 位系统，Windows7 以上

1.3. 依赖库

- None

2. 结构与常量

2.1. 基本类型

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

2.2.1. AFR_FSDK_FACEINPUT

功能描述

脸部信息

定义

```
typedef struct{
       MRECT          rcFace;
       MInt32         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

引擎版本信息.

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
strVersion	字符串形式的版本号
strBuildDate	编译时间
strCopyRight	Copyright

2.3. 枚举

2.3.1. 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

2.3.2. 支持的颜色格式

描述

颜色格式及其对齐规则

定义

ASVL_PAF_I420	8-bit Y 层，之后是 8-bit 的 2x2 采样的 U 层和 V 层
ASVL_PAF_YUYV	Y0, U0, Y1, V0
ASVL_PAF_RGB24_B8G8R8	BGR24, B8G8R8

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 id
SDKKey	[in]	用户申请 SDK 时获取的 id
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                    hMemMgr,
       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

结束引擎

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);
```

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 SDKKEY**，并设置好文件路径和图像尺寸

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <Windows.h>
#include "arcsoft_fsdk_face_recognition.h"
#include "merror.h"

#pragma comment(lib,"libarcsoft_fsdk_face_recognition.lib")

#define WORKBUF_SIZE        (40*1024*1024)
#define INPUT_IMAGE1_PATH "sample1.bmp"
#define INPUT_IMAGE2_PATH "sample2.bmp"
#define APPID       ""                  //APPID
#define SDKKey      ""                  //SDKKey

bool readBmp24(const char* path, uint8_t **imageData, int *pWidth, int *pHeight)
{
	if (path == NULL || imageData == NULL || pWidth == NULL || pHeight == NULL)
	{
		return false;
	}
	FILE *fp = fopen(path, "rb");
	if (fp == NULL)
	{
		return false;
	}
	fseek(fp, sizeof(BITMAPFILEHEADER), 0);
	BITMAPINFOHEADER head;
	fread(&head, sizeof(BITMAPINFOHEADER), 1, fp);
	*pWidth = head.biWidth;
	*pHeight = head.biHeight;
	int biBitCount = head.biBitCount;
	if (24 == biBitCount)
	{
		int lineByte = ((*pWidth) * biBitCount / 8 + 3) / 4 * 4;
		*imageData = (uint8_t *)malloc(lineByte * (*pHeight));
		uint8_t * data = (uint8_t *)malloc(lineByte * (*pHeight));
		fseek(fp, 54, SEEK_SET);
		fread(data, 1, lineByte * (*pHeight), fp);
		for (int i = 0; i < *pHeight; i++)
		{
			for (int j = 0; j < *pWidth; j++)
			{
				memcpy((*imageData) + i * (*pWidth) * 3 + j * 3, data + 
(((*pHeight) - 1) - i) * lineByte + j * 3, 3);
			}
		}
		free(data);
	}
	else
```

```
        {
                fclose(fp);
                return false;
        }
        fclose(fp);
        return true;
}
int main()
{
        /* 初始化引擎和变量 */
        MRESULT nRet = MERR_UNKNOWN;
        MHandle hEngine = nullptr;
        MInt32 nScale = 16;
        MInt32 nMaxFace = 10;
        MByte *pWorkMem = (MByte *)malloc(WORKBUF_SIZE);
        if (pWorkMem == nullptr)
        {
                return -1;
        }
        nRet = AFR_FSDK_InitialEngine(APPID, SDKKey, pWorkMem, WORKBUF_SIZE,
&hEngine);
        if (nRet != MOK)
        {
                return -1;
        }
        /* 打印版本信息 */
        const AFR_FSDK_Version * pVersionInfo = nullptr;
        pVersionInfo = AFR_FSDK_GetVersion(hEngine);
        fprintf(stdout, "%d %d %d %d %d\n", pVersionInfo->lCodebase, pVersionInfo-
>lMajor, pVersionInfo->lMinor, pVersionInfo->lBuild, pVersionInfo->lFeatureLevel);
        fprintf(stdout, "%s\n", pVersionInfo->Version);
        fprintf(stdout, "%s\n", pVersionInfo->BuildDate);
        fprintf(stdout, "%s\n", pVersionInfo->CopyRight);

        /* 读取第一张静态图片信息，并保存到ASVLOFFSCREEN结构体 （以
ASVL_PAF_RGB24_B8G8R8格式为例） */
        ASVLOFFSCREEN offInput1 = { 0 };
        offInput1.u32PixelArrayFormat = ASVL_PAF_RGB24_B8G8R8;
        offInput1.ppu8Plane[0] = nullptr;
        readBmp24(INPUT_IMAGE1_PATH, (uint8_t**)&offInput1.ppu8Plane[0],
&offInput1.i32Width, &offInput1.i32Height);
        if (!offInput1.ppu8Plane[0])
        {
                fprintf(stderr, "fail to ReadBmp(%s)\n", INPUT_IMAGE1_PATH);
                AFR_FSDK_UninitialEngine(hEngine);
                free(pWorkMem);
                return -1;
        }
        offInput1.pi32Pitch[0] = offInput1.i32Width * 3;
        AFR_FSDK_FACEMODEL faceModels1 = { 0 };
        {
                AFR_FSDK_FACEINPUT faceInput;
                //第一张人脸信息通过face detection\face tracking获得
                faceInput.lOrient = AFR_FSDK_FOC_0;//人脸方向
                //人脸框位置
                faceInput.rcFace.left = 346;
```

```
			faceInput.rcFace.top = 58;
			faceInput.rcFace.right = 440;
			faceInput.rcFace.bottom = 151;
			//提取第一张人脸特征
			AFR_FSDK_FACEMODEL LocalFaceModels = { 0 };
			nRet = AFR_FSDK_ExtractFRFeature(hEngine, &offInput1, &faceInput, 
&LocalFaceModels);
			if (nRet != MOK)
			{
				fprintf(stderr, "fail to Extract 1st FR Feature, error 
code: %d\n", nRet);
			}
			/* 拷贝人脸特征结果 */
			faceModels1.lFeatureSize = LocalFaceModels.lFeatureSize;
			faceModels1.pbFeature = (MByte*)malloc(faceModels1.lFeatureSize);
			memcpy(faceModels1.pbFeature, LocalFaceModels.pbFeature, 
faceModels1.lFeatureSize);
		}
		/* 读取第二张静态图片信息，并保存到ASVLOFFSCREEN结构体 （以
ASVL_PAF_RGB24_B8G8R8格式为例） */
		ASVLOFFSCREEN offInput2 = { 0 };
		offInput2.u32PixelArrayFormat = ASVL_PAF_RGB24_B8G8R8;
		offInput2.ppu8Plane[0] = nullptr;
		readBmp24(INPUT_IMAGE2_PATH, (uint8_t**)&offInput2.ppu8Plane[0], 
&offInput2.i32Width, &offInput2.i32Height);
		if (!offInput2.ppu8Plane[0])
		{
			fprintf(stderr, "fail to ReadBmp(%s)\n", INPUT_IMAGE2_PATH);
			free(offInput1.ppu8Plane[0]);
			AFR_FSDK_UninitialEngine(hEngine);
			free(pWorkMem);
			return -1;
		}
		offInput2.pi32Pitch[0] = offInput2.i32Width * 3;
		AFR_FSDK_FACEMODEL faceModels2 = { 0 };
		{
			AFR_FSDK_FACEINPUT faceInput;
			//第二张人脸信息通过face detection\face tracking获得
			faceInput.lOrient = AFR_FSDK_FOC_0;//人脸方向
			//人脸框位置
			faceInput.rcFace.left = 122;
			faceInput.rcFace.top = 76;
			faceInput.rcFace.right = 478;
			faceInput.rcFace.bottom = 432;
			//提取第二张人脸特征
			AFR_FSDK_FACEMODEL LocalFaceModels = { 0 };
			nRet = AFR_FSDK_ExtractFRFeature(hEngine, &offInput2, &faceInput, 
&LocalFaceModels);
			if (nRet != MOK)
			{
				fprintf(stderr, "fail to Extract 2nd FR Feature, error 
code: %d\n", nRet);
			}
			/* 拷贝人脸特征结果 */
			faceModels2.lFeatureSize = LocalFaceModels.lFeatureSize;
			faceModels2.pbFeature = (MByte*)malloc(faceModels2.lFeatureSize);
			memcpy(faceModels2.pbFeature, LocalFaceModels.pbFeature, 
faceModels2.lFeatureSize);
		}
		/* 对比两张人脸特征，获得比对结果 */
		MFloat  fSimilScore = 0.0f;
		nRet = AFR_FSDK_FacePairMatching(hEngine, &faceModels1, &faceModels2, 
&fSimilScore);
		if (nRet == MOK)
		{
			fprintf(stdout, "fSimilScore =  %f\n", fSimilScore);
		}
		else
		{
			fprintf(stderr, "FacePairMatching failed , errorcode is %d \n", 
nRet);
		}
		/* 释放引擎和内存 */
		nRet = AFR_FSDK_UninitialEngine(hEngine);
		if (nRet != MOK)
		{
			fprintf(stderr, "UninitialFaceEngine failed , errorcode is %d \n", 
nRet);
		}
		free(offInput1.ppu8Plane[0]);
		free(offInput2.ppu8Plane[0]);
		free(faceModels1.pbFeature);
		free(faceModels2.pbFeature);
		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。